Contents

Editorial Policy for *Ada User Journal* 194
Editorial 195
News 197
Conference Calendar 232
Forthcoming Events 239

Articles

J-C Mahieux, B Maudry, A Foster
“Using CORBA to Bring New Life to Legacy Ada Software: an Experience Report” 244

J Klein, D Sotirovski
“The Publisher Framework” 248

Ada-Europe 2006 Sponsors 256

Ada-Europe Associate Members (National Ada Organizations) Inside Back Cover
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

While this issue completes volume 27 of the Journal (gosh, what a long history the AUJ begins to have!) and the year 2006 comes to an end, I come to think that we have plentiful reasons to look forward to a new year 2007 reach with important achievements for the Ada community at large. Let me mention just a few, even before talking about the contents of this issue.
To begin with, early in January 2007 we should be hearing the very final word from the ISO top-level governing bodies that the Amendment to ISO/IEC 8652 (a.k.a. the Ada 2005 standard) has been definitely approved: that will be a great achievement, really, which lots of valuable people have contributed to, most of all the ARG team and editor. We have all reasons to expect very good news from that front. Secondly, in deftly planned sync with the progress of the ISO-related events, the “book” (a.k.a. the Springer LNCS edition of the Ada 2005 Reference Manual), generously produced by Ada-Europe, should be starting to arrive at your doorstep. That also is a jolly good news, and a nice gift too. Thirdly, the coming Spring will see a new edition of the IRTAW series, this time focused on drawing the early lessons learned from the use of the very rich set of novel real-time programming features that have come along in the Ada 2005 standard. Even those who are not too attracted to or familiar with real-time systems issues may look forward to the proceedings of IRTAW-13, which will expectedly position on solid ground, Ada 2005 as a leading language for that domain.

It is a few years now that the Ada-Europe annual Conference has introduced the notion of “industrial track”. I am very happy that the proceedings of that session of the conference are published, in a staggered fashion, in successive issues of the Ada User Journal. In keeping with this plan, this issue includes two articles that draw from industrial-track presentations made at the 2006 Conference in Porto. One article, by J-C Mahieux, B Maudry, A Foster, all of PrismTech (a French company specializing in middleware products and solutions) report on a successful use of CORBA technology to migrate a sizeable Ada application to a new target platform. The other article, by J. Klein of Lockheed Martin, and D. Sotirovski of Raytheon Canada, presents the foundation of a distributed object-oriented framework used as the basis of the design of a critical Air Traffic Control application. Both presentations attracted consolidorable interest at the conference.

The rest of the issue contains the usual wealth of news and events of relevance and interest to the Ada community. I am not quite sure the readers appreciate the level of effort that goes in the selection, weaving and editing of that information so that you can have on the journal. You would be surprised should you know the truth! For this reason I am (and you should too, trust me) truly grateful to Santiago Urueña and Dirk Craynest, our News and Calendar editors for their efforts and wish them both (and to all readers too, of course) the best for the new year 2007.

Tullio Vardanega
Padova
December 2006
Email: tullio.vardanega@math.unipd.it
Dear Members of the Ada Community:

On Wednesday, 15 November 2006, the 2006 SIGAda Awards will be presented in a special morning plenary session at the SIGAda 2006 conference in Albuquerque, New Mexico. (See http://www.acm.org/sigada/conf/sigada2006/ if you have somehow missed announcements of this year's annual SIGAda international conference.)

We welcome your nominations of deserving recipients.

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

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

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

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

Please visit http://www.acm.org/sigada/exec/awards/awards.html#Recipients and peruse the names of past winners. This may help you think about the measure of accomplishment that is appropriate. You may be aware of people who have made substantial contributions that have not yet been acknowledged. Nominate them.

Consider what you believe to be the best developments in the Ada community or SIGAda in the last year; the last 5 years; since Ada's inception. Who was responsible? Nominate them.

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

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

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

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

Thank you,

John McCormick, Chair ACM SIGAda

[See also “Nov 12–16 — SIGAda 2006 Conference” in this issue. —su]

AdaCore's Ben Brosgol receives SIGAda's Outstanding Ada Community Contribution Award

Wednesday November 22, 2006

Ben Brosgol, a senior member of AdaCore’s technical staff, received an Outstanding Ada Community Contribution Award at the ACM SIGAda 2006 Conference in Albuquerque, New Mexico (US), on 15 November. This annual award is bestowed on individuals “for broad, lasting contributions to Ada technology and usage”, and past recipients from AdaCore are Robert Dewar (1995), Ed Schonberg (1997), Bob Duff (2002), and Matthew Heaney (2005).

In honoring Ben, SIGAda noted his numerous and significant contributions to the Ada effort, starting from the earliest days and continuing to the present. He worked on the “Red” language (the runner-up in the original Ada language design competition) and served as a Distinguished Reviewer for Ada 83 and as a member of the language revision team for Ada 95. He has been conducting Ada courses for over 20 years, has presented papers and tutorials at many Ada-Europe and SIGAda conferences, and is currently the President of the Ada Resource Association.

In 1998 Ben received SIGAda’s other annual award, for distinguished service to SIGAda itself; he is one of only three individuals to have received both awards.

12-16 November — SIGAda 2006 Conference

Join us this year in Albuquerque, New Mexico from November 12th through the 16th for the annual SIGAda Conference. This year's program includes two and a half days of technical presentations from researchers, academia and industry. Topics include the integration of Ada 2005 into Visual Studios 2005, issues for safety critical and high-integrity systems, using Ada in introductory Computer Science courses, etc. Our conference also includes two days of outstanding tutorials led by some of the most respected technical leaders in the industry. These tutorials range from introductory topics in Ada programming to advanced topics in the new Ada 2005 standard and .NET programming.

Ada User Journal Volume 27, Number 4, December 2006
We have three outstanding keynote speakers this year.
Judith Klein, Lockheed Martin - Use of Ada in Lockheed Martin for Air Traffic Management and Beyond.
Tucker Taft, SoftCheck - Why You Should be Using Ada 2005 now!
You can find more detailed information in the Advance Program on the conference web site:
http://www.acm.org/sigada/conf/sigada2006/
Greg Sicca and Ricky E. Sward, SIGAda Conference Co-Chairs
[See also “15 Nov — SIGAda Award Nominations” in AUJ 27-3 (Sep 2006), p.134. —su]

Feb 24–25 — FOSDEM 2007
From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Ada at FOSDEM 2007 - call for participants
Date: 6 Oct 2006 03:45:42 -0700
Newsgroups: comp.lang.ada
The Ada day at FOSDEM 2006 was successful, how about doing something similar in 2007 again? In particular, I think an introduction to Ada 2005 would be a good idea.
http://www.fosdem.org is down due to hardware failure (this happened to the Ada-France server this year, too), but the mailing list still exists. No planning seems to have taken place yet, so if we want to reserve a room for our presentations, now is a good time. I assume that FOSDEM 2007 is still planned for late February 2007, presumably 24–25 February. As usual, this will be in Brussels, Belgium.
I could redo my classic "Ada in Debian" speech, but I do not have much more to say than I did last year, so I will only speak if there is popular demand for it. OTOH, Debian is due to be released on Dec 4, so by February 2007 I’ll be in the planning phase for Etch1, which might lead to interesting discussions.
Let’s assume that the traditional, slot-based schedule will be applicable this year again, i.e. there are six one-hour slots in a day, and we need to allow for 10 minutes of transition between slots. So, all presentations should be formatted to a 50-minute schedule including questions and answers. […]
In 2006, the "Ada day" was on Sunday. In 2007, if we’re early enough we might get to choose between Saturday and Sunday, or even get a room for the whole two days! (but this requires more speakers).
Besides speeches, we could also have:
- programming tutorials (e.g. have fun with distributed systems)
- a programming contest ("obfuscated Ada", perhaps? :)
- hands-on demonstrations
- unconstrained questions and answers
At this point, I would like to have a "gut feeling" on whether or not an "Ada day at FOSDEM" is feasible or desirable. This "gut feeling" should be sufficient for me to ask for a developer’s room for 1/2 day, 1 day, 3/2 days, or 2 days. It would be nice to have that by the end of October. Later on we can agree on a precise schedule, which must be finalised by mid-November.
If you would like to speak at FOSDEM, attend, or if you simply would like to know what’s in the works, please subscribe to the AdaFOSDEM mailing list. This list is not moderated, but only subscribers can post and browse the archives.
From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Ada at FOSDEM 2007 - call for participants
Date: 24 Oct 2006 04:55:48 -0700
Newsgroups: comp.lang.ada
The dates for FOSDEM 2007 have just been announced on http://www.fosdem.org. The site is now back up on-line. As I suspected the dates are 24–25 February 2007 in Brussels, Belgium.
Two people have already offered presentations; we need more! Please subscribe to the AdaFOSDEM mailing list if you are interested.
[See also “Feb 25–26 — FOSDEM 2006” in AUJ 26-4 (Dec 2005), p.231. —su]

From: dirk@apollo.cs.kuleuven.ac.be (Dirk Craeynest)
Subject: Ada-Europe 2007 submission deadline approaching
Date: 9 Nov 2006 21:15:13 +0100
Organization: Ada-Europe
Newsgroups: comp.lang.ada,fr.comp.lang.ada,comp.lang.misc
12th International Conference on Reliable Software Technologies — Ada-Europe 2007
25 – 29 June 2007, Geneva, Switzerland
Organised, on behalf of Ada-Europe, by École d’Ingénieurs de Genève in cooperation with ACM SIGAda (approval pending)
Ada-Europe organizes annual international conferences since the early 80’s. This is the 12th event in the Reliable Software Technologies series, previous ones being held at Montreux, Switzerland (’96), London, UK (’97), Uppsala, Sweden (’98), Santander, Spain (’99), Potsdam, Germany (’00), Leuven, Belgium (’01), Vienna, Austria (’02), Toulouse, France (’03), Palma de Mallorca, Spain (’04), York, UK (’05), Porto, Portugal (’06).

General Information
The 12th International Conference on Reliable Software Technologies (Ada-Europe 2007) will take place in Geneva, Switzerland. Following the usual style, the conference will span a full week, including a three-day technical program and vendor exhibitions from Tuesday to Thursday, along with parallel workshops and tutorials on Monday and Friday.

Topics
In the last decade the conference has established itself as an international forum for providers and practitioners of, and researchers into, reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the design, development and maintenance of long-lived, high-quality software systems for a variety of application domains. The program will allow ample time for keynotes, Q&A sessions, panel discussions and social events. Participants will include practitioners and researchers from industry, academia and government organizations interested in furthering the development of reliable software technologies. To mark the completion of the technical work for the Ada language standard revision process, contributions that present and discuss the potential of the revised language are particularly sought after.
For papers, tutorials, and workshop proposals, the topics of interest include, but are not limited to:
- Methods and Techniques for Software Development and Maintenance: Requirements Engineering, Object-Oriented Technologies, Formal Methods, Re-engineering and Reverse Engineering, Reuse, Software Management Issues
- Software Architectures: Patterns for Software Design and Composition, Frameworks, Architecture-Centered Development, Component and Class Libraries, Component-Based Design
- Enabling Technology: CASE Tools, Software Development Environments and Project Browsers, Compilers, Debuggers and Run-time Systems
- Software Quality: Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems
Ada-related Resources

Ada & Software Engineering Library and CD-ROM

September 7, 2006

A searchable on-line copy of the Ada and Software Engineering Library, the on-line version of the famous ASE CD-ROMs, has been created at the AdaIC. (The previous home of this material has gone off line.) The library is also available via FTP at Ada-Belgium’s site:


[See also same topic in AUJ 21-4 (Jan 2001), p.224. —su]

Ada-related Tools

Simple components

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Subject: ANN: Simple components v2.4
Date: Sun, 8 Oct 2006 14:53:09 +0200
Newsgroups: comp.lang.ada

Changes:

1. Doubly-linked webs and lists of items with referential semantics were added. Items can be of any type, including tasks, protected objects, unconstrained strings etc;
2. Get_Line procedure was added to the abstract source interface to improve parser performance in the cases when compiler optimization is poor;
3. Slicing and concatenation operations were added to the package Object.Handle.Generic_Bounded_Array;
4. The code was slightly re-arranged to circumvent bugs of GNAT 2006, GCC 4.1.1 (20060525).

[See also same topic in AUJ 27-3 (Sep 2006), pp.134–135. —su]
(though large programs do tend to run quite slowly under Valgrind).

I also toyed with oprofile a while back and did not find it as useful (in the case I was working with) as Valgrind/cachegrind but I did not spend enough time with it to be sure.

From: Emmanuel Briot  
Subject: Re: Profiler?  
Date: Thu, 23 Nov 2006 10:29:35 +0100  
Newsgroups: comp.lang.ada

One additional one which I find even more convenient is sysprof on Linux, since it will monitor the whole system, you can start it whenever you want (i.e. you don't have to profile since the start up of your application), and has a nice GUI to examine the results

[See also "Profiling GNAT programs with gprof" in AUJ 26-3 (Sep 2005), p.154. — su]

GNU Ada Compiler

From: Martin Krischik  
Subject: [gnuada] SuSE 10.1 i686 released.  
Date: Tue, 26 Sep 2006 20:02:15 +0200  
Newsgroups: comp.lang.ada

The SuSE 10.1 i686 version of The GNU Ada Tool-chain has been released. You can find details here:
http://gnuada.sourceforge.net/pmwiki.php

The 32bit version is late as the 32bit SuSE system is not as readily available to me. As such I would welcome it if someone would take the 32 bit release of me. It is not that much work once build environment has been set up.

[See also same topic in AUJ 27-3 (Sep 2006), p.135. — su]

GNU Ada GPS

From: Martin Krischik  
Subject: [gnuada] Finaly: GPS 4.0.0 available.  
Date: 12 Sep 2006 04:02:10 -0700  
Newsgroups: comp.lang.ada

There is no official announcement of GPS 4.0 for non Pro customers — so if you like to read up on what GPS 4.0 can do better then your current GPS read the Wikipedia article:

From: Bjorn Persson  
Subject: [gnuada] Finaly: GPS 4.0.0 available.  
Date: Mon, 11 Sep 2006 17:57:56 GMT  
Newsgroups: comp.lang.ada

Alas, it looks like there won't be a GPS package for Fedora anytime soon. It just displays the splash screen and then crashes on a failed assertion.

[See also same topic in AUJ 27-3 (Sep 2006), p.135. — su]

VAD 6.3 — Visual Ada Developer

From: Stephane Richard  
Subject: [gnua][nu] Announcement on behalf of Leonid Dulman  
Date: Thu, 07 Sep 2006 19:56:05 GMT  
Newsgroups: comp.lang.ada

[Leonid Dulman] announces the latest version of his VAD (Visual Ada Developer) application. Version 6.6 has now arrived and features some great things.

Have a look for yourself, right here:
http://websamba.com/guibuilder  


AutoIT — Automated GUI Testing

From: Per Sandberg  
Subject: [ANN] ada-AutoIT 0.5.2 Released  
Date: Thu, 23 Nov 2006 06:49:34 +0100  
Newsgroups: comp.lang.ada

http://sourceforge.net/projects/ada-autoit/  
Release: 0.5.2  
Date: 2006-11-22  
Changes since last release:  
* Updated to autoIT 3.2  
* Added HTML documentation (Extracted from .chm file) to be used in GPS  
* Added GPS integration  
* Changed install method to make script  
* Included Release notes in Installation AutoIT is a scripting-language/tool for GUI automation.

GUI packages for Ada

From: Stephen Leake  
Subject: Re: GWInows  
Date: Thu, 12 Nov 2006 02:04:25 -0500  
Newsgroups: comp.lang.ada

> I've never seen any reference to a Linux port of GWInows myself.  
I certainly hope not.

The original rational for GWInows was to be a rational Ada binding to the Microsoft Windows API.

Changing that to some other OS/GUI combination would be just wrong.

Which windowing API do you want on GNU/Linux?

From: Stephen Leake  
Subject: Re: GWInows  
Date: Thu, 12 Nov 2006 02:04:25 -0500  
Newsgroups: comp.lang.ada

> Probably GTK+, because that's what I know from GtkAda - ). More important it is widely used and plain C, which may make it easier to bind to than something written in C++. But if there is a better choice I wouldn't object.

Though GTK+ performs quite poorly on Windows platform. And overall, when its documentation tells you that you fundamentally cannot save and restore the position of a window, what could you say?

In my opinion it must be 100% Ada. I don't believe in C.

From: Jeffrey Creem  
Subject: Re: GWInows  
Date: Fri, 10 Nov 2006 23:40:23 +0100  
Newsgroups: comp.lang.ada

I've never seen any reference to a Linux port of GWindows myself.

> But not making the GUI lib portable makes it useless for projects targeting more than just Windows. Obviously.

So if you require portability across platforms, don't use GWindows.

Volume 27, Number 4, December 2006 Ada User Journal
On the other hand, making the GUI library portable means using only those features common to all of the targeted platforms.

So if you want your application to be able to take full advantage of the Win32 API, use GWindows.

**Erlang/Ada Interface**

From: Samuel Tardieu <sam@rfc1149.net>
Subject: Re: PolyORB - building and applications
Date: 21 Sep 2006 15:32:49 +0200
Newsgroups: comp.lang.ada

> I recall someone mentioning an Ada program working as an Erlang node. So perhaps you could leave the communication to an Erlang system?

http://www.rfc1149.net/devel/adaerl

**SecurePolyORB — CORBA Common Secure Interoperability v2**

From: vgodunko@rostel.ru
Subject: ANNOUNCE: SecurePolyORB
Date: 16 Oct 2006 02:05:06 -0700
Newsgroups: comp.lang.ada

SecurePolyORB, an implementation of CORBA Common Secure Interoperability version 2, now available for download from Ada-RU site:

http://www.ada-ru.org/files/securepolyorb-0.2w.tar.gz

For now it provides:
- support for SSL/TLS transport mechanism (authentication, encryption and integrity control);
- support for GSSUP (user/password) attribute layer authentication mechanism;
- identity assertion;
- delegation with backward trust evaluation.

**Avatox — Ada To XML**

From: Marc A. Criley <mc@mckae.com>
Subject: Announce: Avatox 1.2 Now Available
Date: Sun, 24 Sep 2006 20:09:02 -0500
Newsgroups: comp.lang.ada

Avatox (Ada, Via Asis, To Xml) is an application that traverses an Ada compilation unit and outputs the ASIS representation of that unit structured as an XML document.

Version 1.2 now supports UTF-8 character encoding, as well as switches to retain ASIS tree files that pre-existed or are generated on the fly, and to "krunch" the XML output into a continuous, non-indented stream.

Avatox 1.2 is available at www.mckae.com/avatox.html.

From: Marc A. Criley <mc@mckae.com>
Organization: McKae Technologies

**Casbah — Ada Wikis**

From: Peter H.M. Brooks@gmail.com
Subject: Wiki written in Ada?
Date: Thu, 14 Sep 2006 14:28:20 +0100
Newsgroups: comp.lang.ada

I know that there are wikis about Ada, I just wondered (I can’t find a reference) if there’s a project to produce a wiki that’s written in Ada.

In particular, I’m interested to know if there’s a distributed wiki model being developed in Ada.

From: Marius Amado-Alves
<marius@amado-alves.info>
Subject: Re: Wiki written in Ada?
Date: Thu, 14 Sep 2006 14:28:20 +0100
Newsgroups: comp.lang.ada

That would be the Casbah, available at http://www.softdeveloop.org/software/ (still in the old “software” directory, waiting for man-hours to update the site.)

It's Miai Certified progressive software. A Casbah system is online at http://www.liacc.up.pt/cgi-bin/casbah/casbah.cgi

From: Peter H.M. Brooks@gmail.com
Subject: Re: Wiki written in Ada?
Date: 10 Sep 2006 18:31:25 -0700
Newsgroups: comp.lang.ada

PHP is quick to write, fairly easy to understand and has intuitive interfaces to the front and back ends. I’d say that it really was the nature of PHP that brought about the idea of a wiki in the first place. A testimony to the ease of writing in PHP is the large number of separate wiki engines that have been written in it (relative to the smaller number in things like C and Python).

To my mind, an Open Source Ada wiki would be just as cross-platform now, much safer to extend and much, much more robust. So it would be a good project. The overall structure of wikis has become clearer now, so to draw up a spec. based on the limits of current implementation would be a useful job all on its own.

From: Brian May
<bam@snoopy.apana.org.au>
Subject: Wiki written in Ada?
Date: Sat, 16 Sep 2006 09:17:27 +1000
Newsgroups: comp.lang.ada

> PHP seems to be the language of choice these days for web applications. [...] I think the speed and robustness of a compiled language would be ideal for any large scale web application. Not an interpreted language like PHP.

PHP is perceived as being quickier to write.

Maybe for small and very simple projects this might be true.

However, as the code size goes up, productivity goes down, and risk of security problems goes up. After you factor in time wasted due to debugging security breaches on a web server and not getting anywhere, what virtual host did the attacker break into? How did an attacker run wget on this system? How did the attacker execute the IRC server after downloading it? Did the attacker do any other damage? [...]

I think it would be an interesting experiment to rewrite something like Mediawiki in Ada + AWS.

From: Marius Amado-Alves
<marius@amado-alves.info>
Subject: Wiki written in Ada?
Date: Tue, 19 Sep 2006 10:51:12 +0100
Newsgroups: comp.lang.ada

Yes. This is not opposed to what I said. The thing is code size goes up very quickly. For me 1000 lines is already an indicator of “you should be coding in Ada”—and not the large number (1000000?) seen on some ads. Of course here we are using “code size” more as a symptom of reliability requirements and the corresponding code complexity. Also efficiency requirements. Clearly requirements for a good wiki. In my lab sometimes we use Moodle (PHP) and the Casbah (Ada) for the same function (writing something or collecting data collectively). The Casbah beats Moodle in speed hands down.

**EWS — Embedded Web Server**

From: Simon Wright
<simon@pushface.org>
Subject: Re: programming a web with Ada
It would want to make a web page in Ada

You can create a web server in Ada using AWS (at http://libre.adacore.com), which offers lots of standard protocols, or my embedded web server (at http://embedded-srvr.sf.net/) which is very minimalist by comparison. May be (probably are) others.

[As can be read in the home page "EWS is a web server construction kit, designed for embedded applications using the GNAT Ada compiler. The Embedded Web Server is designed for use in embedded systems with limited resources (eg, no disk). It supports both static (converted from a standard web tree, including graphics and Java class files) and dynamic pages. It is written in GCC Ada." —su]

From: Simon Wright
"<simon@pushface.org>
Subject: Re: programming a web with Ada
Date: Wed, 18 Oct 2006 05:59:11 +0100
Newsgroups: comp.lang.ada

> Are there any up-to-date utilities for converting a C *.h file into a Ada wrapper package?

You might try the prototype SWIG 'ada' and 'gnat' modules.

These attempt to generate Ada bindings to both C and C++ libraries. The 'ada' module is for any Ada compiler, and produces bindings based on a 'proxy' approach. The 'gnat' compiler targets the GNAT family of compilers, and produces Ada types and objects which are the binary equivalent of their corresponding C/C++ types.

> I don't care if it requires cleaning up by hand afterwards.

The generated bindings are pretty rough, and generally need to be 'pretty print' formatted, by gnapp or another tool. There are also many style 'wants', which should eventually be cleared up.

> Or is it considered better to write the entire wrapper by hand?

Bindings done by hand tend to be better than the auto-generated ones. SWIG produces very thin bindings. A decent compromise is to use SWIG to build a thin binding, and then write a thick binding by hand, on top of the thin.

> (considering a number of C libraries that I would like to be able to access from Ada here but with no native Ada bindings yet).

There is an example of a few SWIG 'gnat' bindings to the GNU Scientific Library (GSL) at
svn co svn://58.163.88.116/anvil/gsl
SWIG with the 'ada' & 'gnat' modules is available via
svn co https://svn.sourceforge.net/svnroot/gnuada/trunk/projects/swig-1.3.29

The repository code is a little out of date, but recent changes should be committed within the next few days.

From: rodkey@dodo.com.au
Subject: Re: C to Ada
Date: 25 Nov 2006 15:52:47 -0800
Newsgroups: comp.lang.ada

> I am still not clear on the difference — what is the 'proxy' approach?

With the 'proxy' approach, the Ada type holds only a pointer to the wrapped C++ object. When an object of the the Ada type is constructed, a corresponding C++ object is created, and its pointer stored in the Ada type object. All operations on the Ada object are then relayed to its internal C++ object.

The 'gnat' binary approach produces an Ada record layout which is equivalent to the C++ class layout. Operations act directly on the Ada object.

> Can I assume that the GNAT target uses GNAT specific features and won't work without GNAT??

Yes, the 'gnat' SWIG module produces bindings which require a GNAT compiler. Perhaps a similar approach (binary-compatible) for other compilers might be attempted, after the existing modules have matured a little more.

From: Jeffrey Creem
"<jeff@thecreems.com>
Subject: Re: C to Ada
Date: Sun, 26 Nov 2006 13:12:21 -0500
Newsgroups: comp.lang.ada

> What's the status of the Ada SWIG support, and where can I find it? I tried Google, but it wasn't clear to me from what I found whether such support is available yet.

It is currently being worked outside of the SWIG tree. The GNAda project at SourceForge setup an area in the SVN tree so that if the current people working on it abandon it, we don't lose the progress.

The hope is that it will soon be mature enough that it can be accepted into the SWIG tree.

From: Jeffrey Creem
"<jeff@thecreems.com>
Subject: Re: Ada (GNAT) and GNU Scientific Library
Date: Sat, 18 Nov 2006 08:48:40 -0500
Newsgroups: comp.lang.ada

> There were also announcements of an ongoing effort to produce a SWIG module, however my understanding is that is not yet finished (and IIRC it aims at C++ bindings). Meanwhile you can try throwing cbind at the task — it may help produce some bindings at least for the needed functions.

Actually, SWIG will bind to more than just C++. It really is a N-way language binding generator. I’ve not checked the progress recently (the work is being done in the SVN repository of of the GNAda project on SourceForge).

Clbind

From: george@gentoo.org
Subject: Re: Ada (GNAT) and GNU Scientific Library
Date: 18 Nov 2006 01:41:35 -0800
Newsgroups: comp.lang.ada

On a related note, I recently "resurrected" the cbind package. It was one of the packages we had in portage (that's Gentoo) and which I did not, at the time, update yet. The usual search however did not turn up any update information. In fact all the references I found were pointing at the location which now look dead (If anybody can give me any pointers to anything "official" I'd be grateful to hear of course). Fortunately we still had the sources on our mirrors, so I somewhat cleaned them up and repackaged (I don't remember all the details right now, minor stuff mostly. Most notably, I converted some script from csh to bash, in order not to force tshell dependency just for one 20-liner. The rest was trivial IIRC (not that even that was hard :))

If anybody is interested to have it you can get the repackaged sources here:
http://dev.gentoo.org/~george/src/cbind-6.0.tar.bz2

or on any of our mirrors. Just run make to build, it only needs GCC and make

AFAICT, I intend to keep it for as long as I can "maintain" it. I briefly tested it on some c file - incidentally that was my attempt to test some gsl function ;), and it seemed to spit some reasonably looking code for some gsl header file. Although I did not test this further at the time.

Cairo Ada binding

From: Damien Carbone
"<damien.carbone@free.fr>
Subject: Re: Ada (GNAT) and GNU Scientific Library
Newsgroups: comp.lang.ada

or on any of our mirrors. Just run make to build, it only needs GCC and make

AFAICT, I intend to keep it for as long as I can "maintain" it. I briefly tested it on some c file - incidentally that was my attempt to test some gsl function ;), and it seemed to spit some reasonably looking code for some gsl header file. Although I did not test this further at the time.
I have a working (at least for all tests I have done with it!) Cairo Ada 95 binding. As soon as Cairo people are OK with it (from Cairo's viewpoint), I expect to publish it completely on Cairo site. Till then, I have put its user API here: http://damien.carbonne.free.fr/cairoada/index.html

All comments will be welcome.

From: Seth Brutzman
<seth.bruzman@gmail.com>
Subject: Re: Cairo Ada binding
Date: Thu, 26 Oct 2006 23:14:20 +0200
Newsgroups: comp.lang.ada

> What's Cairo?
Cairo is a slick 2D graphics library (not a GUI toolkit). Information can be had at http://www.cairographics.org.

Interesting things to note are that it is cross-platform, can output to several different formats including SVG and PDF, and is well on its way to being hardware accelerated through OpenGL.

Downside? It's written in C. ;)

From: Damien Carbonne
<damien.carbonne@free.fr>
Date: Thu, 26 Oct 2006 23:14:20 +0200
Subject: Re: Cairo Ada binding
Newsgroups: comp.lang.ada

> Providing a .tar.gz would be good; it's far easier to browse code on my computer than on the web. Especially if it compiles.

You can download current version of the binding here: http://damien.carbonne.free.fr/download/

Some of the possible changes I could make are listed in the TODO file. Using Cairo with GtkAda needs the addition of a small package (Gdk.Cairo) that needs to be written. I expect to put everything soon on Cairo official site.

**GLOBE_3D — 3D Engine**

From: Gautier de Montmollin
<gdmont@hotmail.com>
Date: Sat, 14 Oct 2006 12:12:00 -0700
Newsgroups: comp.lang.ada

GLOBE_3D is an open-source software. It allows an easy and fast real-time display of objects, of any kind, or groups of connected objects like a series of rooms with open doors.

One single source set – without any conditional compilation – for all target platforms and compilers. Works on:
- operating systems: Windows, Linux, Mac OS X
- compilers: GNAT, ObjectAda

More details here:
http://homepage.sunrise.ch/mysunrise/gdm/g3d.htm

Newsletter on demand.

[See also same topic in AUJ 27-2 (Jun 2006), p.72. —su]

**Vim Ada-Mode**

From: Martin Krischik
<krischik@users.sourceforge.net>
Subject: [Vim] Ada language Mode.
Date: Sun, 08 Oct 2006 17:35:34 +0200
Newsgroups: comp.editors,comp.lang.ada

I have created a new language mode for Ada and would like anybody who is interested in Ada and Vim to comment on.

The new mode offers:
* Support for Ada 2005 keywords.
* Improved syntax highlight (Including all standard Pragmas and Attributes).
* User completion (Keyword, Pragmas, Attributes)
* Omni completions (using ctags or gnat xref).
* Tag search (using ctags or gnat xref).
* Unified on-line help (One ada.txt for all).
* Compiler support for GNAT and Dec Ada (using an extensible OO-Design).
* Three different folding mechanisms.
* All function are autoloaded.
* Optimised for Vim 7

The aim is to replace the Ada language mode, which is currently part of the standard run-time, with this new mode.


From: Martin Krischik
<krischik@users.sourceforge.net>
Subject: Vim Ada Mode: first upstream step
Date: Sun, 12 Nov 2006 19:51:00 +0100
Newsgroups: comp.lang.ada

To keep you updated: The new Vim Ada mode did its first upstream step – that is Bram accepted the patches and uploaded them to the ftp server.


[See also same topic in AUJ 27-3 (Sep 2006), pp.139-140. —su]

**Regular Expressions in Ada**

From: Matthias Kistler
<mki@vmx.de>
Subject: Regular Expressions in Ada
Date: Wed, 08 Nov 2006 12:49:50 +0100
Newsgroups: comp.lang.ada

Does anybody know, if it's possible to use regular expressions in Ada 2005? I come from Perl and I'm very interested in Ada but it's useless for me without the possibility of using regular expressions similar to Perl.

I found a GNAT-package providing only a regex-matcher. But I also need a replacer. Otherwise it'd be useless for me.

Does anybody know about regular expressions in Ada 2005? Is there any tutorial? Is there at least an Ada-library? Or can just anybody explain to me, how to use regexes in Ada?

From: Georg Bauhaus
<bauhaus@futureapps.de>
Subject: Re: Regular Expressions in Ada
Date: Wed, 08 Nov 2006 22:14:51 +0100
Newsgroups: comp.lang.ada

They cover a fair bit of what you would do using Perl's expressions and translation operators in a sane way.

The GNAT packages do provide scanning and replacement. There is a tutorial in the package specifications.

Example programs:


#3 uses Unix style regular expressions.
#4 uses SPITBOL regular expressions.

SPITBOL patterns are quite powerful and fast, in fact you can write an entire program just as a pattern. But don't do that.

From: Pascal Obry <pascal@obry.net>
Subject: Re: Regular Expressions in Ada
Date: Wed, 08 Nov 2006 22:49:30 +0100
Newsgroups: comp.lang.ada

You have seen GNAT.Regexp but probably not GNAT.Regpat. The latter does support replacement. As noted by others there is also GNAT.Spitbol.
Player-Ada — Robotic Platform Binding

From: Alex R. Mosteo
<alessandro@mosteo.com>
Subject: Re: [ANN] Player-Ada 2.0.3.0 released
Date: Thu, 26 Oct 2006 19:51:01 +0200
Newsgroups: comp.lang.ada

Player-Ada is a binding for the Player/Stage robotic platform. Player-Ada is a not-so-thin binding to the libplayerc client library that is distributed as part of the Player/Stage multi-robot interface/simulator software.

It currently implements the following interfaces: blobfinder, gps, laser, localize, planner, position2d, simulation.

Binding homepage: http://ada-player.sf.net/

Player homepage: http://playerstage.sourceforge.net/

From: Alex R. Mosteo
<alessandro@mosteo.com>
Subject: Re: [ANN] Player-Ada 2.0.3.0 released
Date: Fri, 27 Oct 2006 10:42:27 +0200
Newsgroups: comp.lang.ada

> > Does the robot have to be connected to the PC (RS232) in order to function?

If your question is if you can flash some program inside the robot microcontroller and completely forget about the serial and external things, I think the answer is no, or not easily. The robot's microcontroller is documented and its OS is upgradeable. I don't think the documentation says enough to replace the OS with something else, but certainly this is not the use the manufacturer has in mind. The intended use for the robot is via serial. You'd also have other problems: the microcontroller has access to wheels and sonars, but not to laser that is a completely isolated entity. Maybe you could use the now unused serial to link these two.

In the other hand, nothing mandates that the client (user) side of the serial is a PC. The protocol is documented so you could use whatever you want. In fact the robot can be purchased with or without embedded PC. Without it, you need either a radio serial, a laptop to put on the robot, or something else. The advantage of a PC is that you have out-of-the box the proprietary software provided, and Player as an open source option.

If you're concerned with a full real-time solution, this is doable: the microcontroller OS is RT and the periods of wheel encoders and sonars feedback are documented. So nothing precludes using some RT-Linux or other RTOS in the client side.

(Actually, these Pioneer robots are popular within the robotic community. You can probably find other PC software interfaces as well. CARMEN comes to mind).

http://playerstage.sourceforge.net/doc/Player-cvs/player/supported_hardware.html

I have used it with Pioneer DX/AT robots equipped with sonar and SICK200 lasers.

From: Alex R. Mosteo
<alessandro@mosteo.com>
Subject: Re: [ANN] Player-Ada 2.0.3.0 released
Date: Fri, 27 Oct 2006 12:03:10 +0200
Newsgroups: comp.lang.ada

> Hardware?

The binding has only been tested in Linux with GNAT, but it's supposed to be pure Ada 95 without using GNAT extensions. Quoting Player FAQ:

"Player runs on pretty much any POSIX platform, including embedded systems (Player has been cross-compiled to run on several ARM- and PPC-based Linux systems). Specifically, Player's requirements are:

* POSIX development environment, with threads (pthreads)
* TCP stack
* A compiler with both C and C++ (we have only tested GCC, but other compilers may work)
* A bash shell, to run the configure script; this implies that Player will not build natively in Windows, though some users have it running under Cygwin, and there are rumors of MinGW builds as well."

If you refer to what robots can be controlled with player:
automatically connects to the remote machine, synchronises the files when necessary, and performs the action. Limiting the number of operations carried out on the remote server significantly reduces the amount of required network and power usage.

Features/Benefits
AdaCore’s GPS Remote Programming feature offers the major benefits of the “one server/multiple clients” solution, including:

* Greater control of the development environment, ensuring that the code that is tested will be exactly the production code that will run.
* Easier installation of node-locked software, and easier sharing of project sources and builds.
* Ability to develop software that is portable from desktop PCs to several platforms.

Pricing and Availability
Remote Programming is available in GNAT Programming Studio (GPS) 4.0 that accompanies the GNAT Pro development toolset. Contact AdaCore for the latest information on pricing and supported configurations.

(sales@adacore.com)

About AdaCore
Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, a modern programming language designed for large, long-lived applications where reliability, efficiency and safety are critical. AdaCore’s flagship product is GNAT Pro, which comes with expert online support and is available on more platforms than any other Ada technology. AdaCore has customers worldwide; see http://www.adacore.com/home/company/customers/ for more information. Use of Ada and GNAT Pro continues to grow in high-integrity and safety-critical applications, including commercial and defence aircraft avionics, air traffic control, railroad systems, financial services and medical devices. AdaCore has North American headquarters in New York and European headquarters in Paris. www.adacore.com

AdaCore — GNATstack

Tuesday October 31, 2006
AdaCore Announces New Software Stack Analysis Tool
GNATstack is an important component of AdaCore’s High-Integrity solution (GNAT Pro HIE), which is an enhanced Ada development environment used for building safety-critical, embedded software applications that require certification. The tool is targeted at system designers creating high integrity and high reliability embedded applications.

“Manually calculating the amount of memory that should be allocated to a memory stack increases the risk that an embedded application will use more memory on the stack than is available, which can result in memory corruption, unpredictable execution, or a fatal system crash,” said AdaCore senior software engineer Jose Ruiz. “GNATstack uses data generated by the compiler to determine the worst-case stack requirements. This output is used to ensure that sufficient memory is reserved for the stack(s), and to guarantee that the software application executes safely.”

About GNATstack
GNATstack calculates the worst-case stack requirements for every stack entry point by performing per-subprogram stack usage as well as control flow analysis. The tool provides an audit trail for the certification of high integrity and high reliability applications, and can detect and display a list of potential problems when calculating the stack requirements, including:

- Indirect (including dispatching) calls: the tool will indicate the number of indirect calls made from any subprogram.
- External calls: the tool displays all the subprograms that are reachable from any entry point that does not have a stack or call graph information.
- Unbounded frames: the tool displays each reachable subprogram that has an unbounded stack requirement. The required stack size depends on the arguments passed to the subprogram.
- Cycles: the tool can detect all the cycles in the call graph.

Availability and Pricing
GNATstack is available in the GNAT Pro HIE package and as an add-on for GNAT Pro. For more information on GNATstack features, please visit http://www.adacore.com or contact AdaCore (sales@adacore.com).

AdaCore — PyGTK

URL: http://www.adacore.com/2006/11/22/pygtk-a-testing-solution-for-gps/
PyGTK: A testing solution for GPS
Wednesday November 22, 2006
The GNAT Programming Studio, AdaCore’s IDE, has been enhanced to allow python scripts that use PyGTK’s interface to the GTK+ toolkit to interact with it.

GPS architecture allows the user to interact with the GUI by means of scripts written in either the simple GPS script language or in python. Current versions of GPS offer the possibility, for example, to open a new source editor and move the cursor to the end of the buffer by writing:

* ed = GPS.EditorBuffer.get (GPS.File("src.adb"))
* ed.current_view().goto( (end_of_buffer())

What PyGTK brings is the ability to simulate user-level actions such as mouse clicks or key strokes, to manipulate complex widgets such as GTK’s TextView and TreeView, to activate contextual menus, etc.

This is a revolution as far as GPS testing is concerned because it allows most actions that previously required human interaction to be completely automated. For example, let’s assume an action opens a dialog containing an OK button. Simulating a mouse click on it is as simple writing:

* ok_button.clicked()

PyGTK allows automatic testing of everything including the most complex GUI aspects such as focus issues, signal handling, etc.

AdaCore — Transition to GCC 4.1 backend

New stage in the transition to GNAT Pro based on gcc 4.1 backend
Friday November 17, 2006
We are in the process of transitioning the GNAT Pro technology to a new compiler back-end based on GCC 4.1 which we expect to bring significant performance improvements to user applications. Our goal is to have several of our supported configurations on this back-end for the next major GNAT Pro release scheduled early 2007. We have been able to make significant progress in the areas of general stability and in the support for numerous platforms thanks to invaluable input provided by our customers. This latest beta version includes support for most of the new Ada 2005 features and is our most advanced Ada 2005 technology.

Please do not hesitate to contact us if you have any questions concerning this program.

AdaCore — Ada 2005 support in gnatpp

Partial support for Ada 2005 features in gnatpp
Tuesday November 7, 2006
A new -gnat05 option is added to gnatpp. When called with this option, gnatpp can process Ada sources containing some Ada 2005 features:
- overriding indicators
- null subprograms
- interface types
- generalized anonymous access types
- null exclusion
- tagged incomplete types
- limited aggregates (e.g. in component associations)
- program calls given in Object.Operation notation
- limited and private with clauses
- raise with string message
- formal abstract subprograms
- partial parameter lists for formal packages

A future gnatpp version will fully support Ada 2005.

AdaControl — New Version Numbering for GNAT Pro

New Version Numbering for GNAT Pro
Friday November 10, 2006

In 2007 AdaCore will be moving to a new numbering scheme for product releases. Instead of two-part version numbers such as 3.15 or 5.04a1, we will be using the more common convention of three numbers separated by dots. The first number, as at present, will identify a major release and will thus indicate the introduction of significant new functionality. The second number will correspond to the digits after the dot in the current scheme. And the third number will replace the suffix (such as “a” or “1”) used at present. “0” as the third number will be used for a beta version, “1” for an initial release, and higher numbers for subsequent releases.

In order to acknowledge the full support for the new Ada 2005 features, the GNAT Pro major version number is moving to the 6 series. More specifically, the version scheduled for Q1 2007 will be 6.0.1, and the follow-up release scheduled for later in the year will be 6.0.2. The planned releases in 2008, incorporating enhancements to be made during 2007, will then be 6.1.1 and 6.1.2.

Customers can find more information in the “Our support policy” on GNAT Tracker.

AdaCore — AdaControl

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: AdaControl V1.5 released
Date: Wed, 11 Oct 2006 19:03:44 +0200
Organization: Adalog
Newsgroups: comp.lang.ada

AdaControl is a commercial product of Adalog; for information about support and assistance with AdaControl or more generally issues related to coding rules enforcement, please write to info@adalog.fr

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: AdaControl V1.6 released
Date: Wed, 06 Dec 2006 16:44:39 +0100
Organization: Adalog
Newsgroups: comp.lang.ada

AdaControl is a commercial product of Adalog and can develop new rules that fit your particular needs; it also provides consultancy about coding standard. If you are interested, please write to info@adalog.fr

Visustin v4 updates its support to the GMGPL license, which allows you to use it, or any part of it, without restrictions.

Adalog provides commercial support for AdaControl and can develop new rules that fit your particular needs; it also provides consultancy about coding standard. If you are interested, please write to info@adalog.fr

Usualy, this kind of announcement lists the improvements over the previous version, but we won’t do it this time; there are too many!

Suffice it to say that AdaControl features now 45 rules, with many sub-rules, making a grand total of 156 different checks available!

Usability has improved a lot too, with a complete integration into GPS.

To learn more about AdaControl, (and download it), go to http://www.adalog.fr/adacontrol2.htm

Parts of the new developments have been supported by EuroControl, BelgoControl and CSEE-Transport.

As always, AdaControl is provided under the GMGPL license, which allows you to use it, or any part of it, without restrictions.

Visustin v4 supports a total of 29 programming languages: Ada, ASP, BASIC, C/C++, C#, Clipper, COBOL, Fortran, Java, JSP, JavaScript, LotusScript, MASM, MSP430, NASM, Pascal/Delphi, Perl, PHP, PL/SQL, PowerScript, PureBasic, Python, QuickBASIC, REALbasic, T-SQL, VB, VBA, VB.NET and Visual FoxPro.

Supported operating systems:
- Windows
- UNIX

Visustin home page: http://www.visustin.com

About Visustin:


Aivosto — Aivosto ADT Eclipse Toolkit

From: Tom Grossman <grossman@aonix.com>
Subject: ANN- AonixADT Ada Development Toolkit for Eclipse v. 3.11
Date: Fri, 15 Sep 2006 17:36:38 +0200
Organization: Aonix
Newsgroups: comp.lang.ada

Aonix ADT is a freely available plugin for Ada language development in Eclipse. In addition to supporting standard Eclipse
functionality and views, AonixADT also provides:
Support for Multiple Ada Compilers and Tools
* ObjectAda toolchain support
* GNAT toolchain support
Ada Project Navigator
* Ada-specific navigation of project files and folders with expansion of files to show internal constructs (variables, subprograms, types, etc.)
* Navigation to and from source code in the Ada Editor.
Ada 95 Colorizer
* Customizable colorization of Ada source code
* Editor support for configurable code indention while new code is written.
* Automatic parenthesis matching, block matching, etc.
* Semantic, project-wide navigation of Ada objects (variables, units, etc.) from editor including opening of the spec and body declaration and searching for references.
* Syntactic and Semantic Code Assist for Ada constructs as well as application objects such as variables, packages, procedures, functions, types, exceptions, and tasks.
Build Automation
* Automatic, incremental builds of projects.
Configuration File
* Storage of all project build properties in ASCII text files which can be put under CM control along with source files.
Navigation to Compilation Errors
* Build errors are displayed in Problems view with navigation to errors in source code.
Navigation to ARM
* Build errors allow easy navigation to relevant section of hypertext Ada 95 Reference Manual for ObjectAda.
Pretty Printing
* Whole file source code reformattting to match project-customizable preferred format.
* Support for gnatpp pretty printer.
Multiple Partitions
* Support projects that contain software for more than one partition. This means the ability to build more than one executable in one project space.
Eclipse Wizards
* Ada Project Wizards
* File, Package, Procedre, Function creation wizards.
Compatibility with Other Languages
* Support for multi-language projects using CDT and other Eclipse plugins.

Configurable Toolchains
* Support for configuration of multiple Ada toolchains.
Multi-language Graphical Debugger
* Support for configuring and debugging of executables and attaching to already running processes.
AonixADTv311 is available at http://www.aonix.com/adt.html.
Aonix is an Add-in Provider Member of the Eclipse Foundation.
From: Tom Grosman <grosman@aonix.com>
Subject: Re: ANN- AonixADT Ada Development Toolkit for Eclipse v. 3.11
Date: Fri, 15 Sep 2006 17:44:33 +0200
Organization: Aonix
Newsgroups: comp.lang.ada
Our website currently has AonixADTv311 for Windows 2000 and XP, with Linux and Solaris versions to be made available shortly.
ObjectAda 8.2+ as well as various flavors/versions of GNAT are supported.
The Quickstart guide available in pdf from the ADT download page should give you a good idea of the functionality and look and feel of ADT.
From: Tom Grosman <grosman@aonix.com>
Subject: [ANN]- AonixADT for Eclipse now available for Intel/Linux and Sparc/Solaris for GNAT
Date: Mon, 30 Oct 2006 18:19:06 +0100
Organization: Aonix
Newsgroups: comp.lang.ada
Aonix is pleased to announce AonixADTv311, the first publicly available release of our Eclipse Ada Development plugin for Intel Linux and Sparc Solaris. In addition to introducing support for Linux and Solaris, version 311 updates AonixADT for Windows by adding support for low-level debugging operations and other enhancements.
AonixADT supports GNAT and ObjectAda toolchains and includes support for Ada Project Navigation
Ada Semantic Browsing
Ada 95 Colorization
Ada Semantic Code Assist
Multilanguage Development
Multiple Build Configurations
File Creation Wizards
Code Formatting
Native Debugging
From: Tom Grosman <grosman@aonix.com>
Subject: Re: ANN- AonixADT Ada Development Toolkit for Eclipse v. 3.11
Date: Fri, 15 Sep 2006 17:44:33 +0200
Organization: Aonix
Newsgroups: comp.lang.ada
> I don't understand is why a Java plugin for an IDE written in Java wouldn't run on all platforms automatically. I understand that the SWT GUI stuff is platform specific, but I would think that would mean just recompiling.

FWIW, JDT and CDT (in addition to Eclipse itself) also have separate plugins for different platforms.

Aonix — ObjectAda for Windows 8.2
From: Owner-Intel-ObjectAda <owner-intel-objectada@aonix.com>
To: intel-objectada@aonix.com
Date: Thu, 12 Oct 2006 17:25:24 -0700
Subject: Re: ANN- AonixADT Ada Development Toolkit for Eclipse v. 3.11


For information on obtaining or renewing a maintenance agreement, please contact your nearest Aonix Sales office. For contact information see http://www.aonix.com/contact_us.html. [See also same topic in AU 27-3 (Sep 2006), p.143. —su]

DDC-I — SCORE for Texas Instrument's DSP
DDC-I Announces Availability of SCORE Integrated Development Environment for TMS320C40 DSP
Provides seamless upward migration path from Ada 83 to mixed Ada 95/Embedded C++ for legacy C40 code.
Phoenix, AZ. December 4, 2006. DDC-I, a leading supplier of development tools for safety-critical applications, today announced the availability of its SCORE® Integrated Development Environment (IDE) for Texas Instrument's TMS320C40. The SCORE IDE makes it easy for C40 developers to take existing Ada 83 programs developed for the C40, upgrade them using a mixture of Ada 95 and Embedded C++, and deploy them on a royalty-free Ada 95 run-time system. The SCORE IDE also makes it easy for C40 developers to migrate their code to other processors such as the PowerPC and X86, with the unique ability to debug multiple targets and languages at the same time.

"There has been a lot of Ada 83 code developed for the C40, particularly in defence applications," said Bob Morris, president and CEO of DDC-I. "SCORE provides a modern, best-in-class mixed language development environment that makes it easy for C40 developers to upgrade their Ada 83 code and take advantage of the latest Ada 95 and Embedded C++ technology. SCORE also makes it easy for developers to migrate existing C40 code to new processors."

To support the C40, DDC-I has developed a new C40 compiler, code generator, and disassembler. The SCORE IDE provides full JTAG multiprocessor debugging for the C40, including trace and the ability to monitor all registers. SCORE also provides a PC-based C40 instruction set simulator.

SCORE® is a mixed-language, object-oriented IDE for developing and deploying safety-critical applications. SCORE provides optimizing compilers for Ada, C, Embedded C++, and Fortran77, all of which pass the applicable ACATS, PlumHall, Perennial, and FCVS compiler validation suites. The SCORE® IDE features an intuitive GUI with industry leading features such as a color-coded source editor, project management support, and automated build/make utilities. SCORE's mixed-language, multi-window, symbolic debugger recognizes C/EC++, Ada and Fortran syntax and expressions, and can view objects, expressions, call chains, execution traces, interspersed machine code, machine registers, and program stacks. The debugger supports full Ada-level debugging, including constraints, attributes, tasks, exceptions, break-on-exception and break-on-tasking events. The debugger is non intrusive, can debug at the source or machine level, and can be enabled without changing the generated code.

SCORE provides versatile run-time target options, including a bare run-time system certifiable to Level A of the FCC DO-178B standard, and an enhanced bare run-time system for simulated and emulated environments. The SCORE IDE is available immediately for the TMS320C40. Pricing starts at $5000.

About DDC-I, Inc.

DDC-I, Inc. is a global supplier of software development tools, custom software development services, and legacy software system modernization solutions, with a primary focus on safety-critical applications. DDC-I's customer base is an impressive "who's who" in the commercial, military, aerospace, and safety-critical industries. DDC-I offers compilers, integrated development environments and run-time systems for C, Embedded C++, Ada, JOVIAL and FORTRAN application development.

McKae Technologies —

XML EZ Out

From: Marc A. Criley <mc@mckae.com>
Date: Sun, 24 Sep 2006 13:54:26 -0500
Subject: Anounce: XML EZ Out 1.05
Available
Newsgroups: comp.lang.ada

XML EZ Out is a small set of packages intended to aid the creation of XML-formatted output from within Ada programs. It basically wraps the tags and data provided to it with XML syntax and writes them to a user-supplied medium.

This medium can be any sort of writable entity, such as a file, a memory buffer, or even a communications link, such as a socket. The only functionality required of the medium is that it supply a meaningful "Put" (for writing a string) and "New_Line" procedure.

Simply "with" the desired package, instantiate it if necessary, and then "use" it. The XML EZ Out packages are explicitly designed to have "use" clauses applied.

Version 1.05 adds a couple features, one to let applications set and change the style of the generated XML, either as indented or continuous, during run-time instead of as a parameter of the instantiation.

The other addition controls the presence of attributes that have no content, i.e, an empty string. By default the attribute is not output in this situation, but setting Default_Output_Null_Attributes to True forces those attributes having empty content to be output. [Suggested by Niklas Holstj.]

Licensing is GMGPL.

www.mckae.com/xmlEZ.html

Ada and GNU/Linux

Ada support in Ubuntu

From: Marc A. Criley <mc@mckae.com>

Organization: McKae Technologies
Subject: Re: ubuntu gcc
Date: Sat, 11 Nov 2006 09:44-0600
Newsgroups: comp.lang.ada

> Does the GCC that comes with Ubuntu GNU/Linux support Ada (with appropriate packages installed)?

Yes, "gnatmake -v" for Ubuntu 6.06 shows:

GNATMAKE 4.0.3 (Ubuntu 4.0-3)

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: ubuntu gcc
Date: Sat, 11 Nov 2006 17:01:28 +0100
Newsgroups: comp.lang.ada

I recommend against it; in Ubuntu 6.06 "Dapper Drake" I recommend using the "gnat" package from universe instead. This is GNAT 3.15p and it comes with the full complement of libraries, like in Debian.

Ubuntu 6.10 "Edgy Eft" contains all the Ada packages transitioned to GCC 4.1.1, also by means of package "gnat", also in universe.

AdaControl Linux package

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: AdaControl V1.5 released
Date: 25 Oct 2006 08:11:20 -0700
Newsgroups: comp.lang.ada

> Adalog is pleased to announce the release of a new version of AdaControl, the free tool for checking Ada programming rules.

And AdaControl 1.5 has now reached testing in Debian, meaning it will be in the next stable release, Etch.

Do other distributions carry AdaControl?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: AdaControl V1.5 released
Date: 26 Oct 2006 01:50:58 -0700
Newsgroups: comp.lang.ada

> Ubuntu Edgy Eft have version 1.4 in Universe.

Well that's my package, since Ubuntu is a derivative of Debian. In fact, the Ubuntu folks asked me to provide gnade 1.6.1 and gnat-glade 2006-3 before it became available in Debian so they could include it in Edgy Eft, which will be frozen Real Soon Now.

Debian build scripts

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Announce: Debian build scripts on a public Monotone server
Date: Mon, 25 Sep 2006 22:18:15 +0200
Newsgroups: comp.lang.ada

As most of you know, all Debian build scripts are public; you can download them.
from Debian's many mirrors, change them, and run them to build binary packages. Now, you can also follow their development almost in real-time thanks to Monotone, a powerful distributed version control system.

About Debian source packages

A Debian source package consists of three files:

* {package}_{version}.orig.tar.gz — the pristine upstream sources. The directory tree in the tarball always starts from {package}_{version}.orig (note: '_' in the tarball name but '-' in the directory name).
* {package}_{version}_{revision}.diff.gz — a compressed patch which applies to the above directory tree extracted from the tarball. This patch brings in all of the Debian build scripts, Debian-specific files, and patches. The {revision} is specific to Debian, too, and changes with every upload. The most important file brought by the patch is debian/rules, which is executable and builds the package. Usually (and always for the Ada packages), debian/rules is a Makefile.
* {package}_{version}_{revision}.dsc — a short text file containing the MD5 sums of the two above files, and which is signed with the maintainer's private GPG key.

In order to build the set of binary packages for a source package, one must therefore:

$ tar xzf {package}_{version}.orig.tar.gz
$ zcat {package}_{version}-
{revision}.diff.gz | patch -p0
$ mv {package}_{version}_{revision}.dsc
$ cd {package}_{version}
$ debian/rules binary

All these steps, and then some, are automated by apt-get:

$ apt-get source --build {package}

So, what's the problem?

With the above system, you can download and rebuild from source any Debian package from your current distribution (stable, testing, unstable, or experimental). You can change the package itself, or the build scripts. But you cannot:

- use older versions of the build scripts
- use newer build scripts that have not yet been published to the Debian archives
- submit your changes to the maintainer in a clean, efficient and automated way (the official way is by opening a bug report).

These problems become critical for the many packages that are maintained by teams rather than individuals.

A version control system allows the development build scripts to become public, so that team maintenance is possible. For the Ada packages, I would like to encourage people to look at the scripts and propose improvements; maybe even form a team and benefit from each other's experience? [...] [See also “Monotone — A Distributed Revision Control System” in this issue — su]

Monotone — A Distributed Revision Control System

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Subject: Announce: Debian build scripts on a public Monotone server

Date: Mon, 25 Sep 2006 22:18:15 +0200

Newsgroups: comp.lang.ada

[See also “Debian build scripts” in this issue — su]

[...] Many Debian packages have project pages on Alioth[1], and use one of Subversion, GNU Arch, Bazaar-NR or GIT as their version control system.

Since I do most of my Debian work on the train and without any network connection, I require a distributed version control system. After evaluating several candidates, I settled on Monotone several months ago.

http://www.ada-france.org/debian/distributed-version-control-systems.html

Why Monotone?

I believe that Monotone is the Ada of version control systems, so it is only appropriate that I use it for my Ada work. Monotone is safe, correct and powerful by design — It uses cryptographic keys to authenticate changes. It is written by elite programmers who, despite using C++, have the "Ada attitude": no pointers, one assert() every 9 lines of code, massive use of generics (templates), and not a single critical bug in 3 years. The slides at [its web page] and my own tests convinced me to switch from Meta-CVS several months ago for my Debian packages, as well as for other work.

A Monotone database consists of one single file; this is very convenient for maintenance. A Monotone database takes only a fraction of the disk space required for an equivalent database in any other system I've tried (Subversion, Bazaar-NR, Mercurial, CVS), which is also an important consideration for me.

(I like to think that CVS is the "C" of version control systems, Subversion is the "C++" designed to replace the "C", GIT is the "assembly language" who needs cogito to be useable, Bazaar-NR is the "perl", grossly inefficient and completely baroque, Mercurial is the "Eiffel" i.e the second best, Monotone is the "Ada", i.e. the best, even if not perfect)

(I particularly dislike Subversion and its distributed derivative, SVK. I do not recommend them because their working model is inherently broken, IMHO. A branch is NOT a directory, and a tag is NEVER a branch NOR a directory. And Subversion does not even try to keep track of merges; just like C++ does not even try to multitask.)

What's in the Ada-France database?

The database that I just published on Ada-France is a replica of the one I work on every day. It contains one branch (sometimes a couple of branches, actually) for each package I work on. Each published upload of each package also has a tag. You can browse the whole history of all changes, with comments. The size of the database is about 1.7 Mb.

As of today, "mnt list branches" says:

org.debian.adacontrol
org.debian.asis
org.debian.asis-doc
org.debian.asis-2005
org.debian.gnat-gdb
org.debian.gnat-glade
org.debian.gnat-gps
org.debian.libaunit
org.debian.libaws
org.debian.libflorist
org.debian.libgtkada2
org.debian.liblinden
org.debian.liblindenoken
org.debian.libtemplates-parser
org.debian.libtexttools
org.debian.libxmlada1
org.debian.libxmlada2

The list of tags ("mnt list tags") would be too boring for this post.

Each branch contains a "debian" directory and, in most cases, a "patches" directory. I use Quilt to manage the patches.

http://savannah.nongnu.org/projects/quilt

If you extract an upstream source tarball, and checkout from Monotone into the source tree, you're ready to build the package.

The database does not contain the upstream tarballs (.orig.tar.gz); these are available from Debian mirrors anyway or, if not yet published in the Debian archive, from http://www.ada-france.org/debian/pool.

How to use Ada-France's Monotone database

If you want to *read* from the database:

1. Install Monotone, version 0.26 or later (the server is currently running 0.28).
2. Create a key pair:

   $ mtn genkey your@email.address

3. Create a new, local database:

   $ mtn --db=debian.mtn db init

   (I like to keep all my databases in /var/lib/monotone, but you can place your database anywhere; remember: it's a single file anyway. You can, of course, have as many databases as you want.)
4. Pull all branches starting with "org.debian":

   $ mtn pull org.debian
Thanks for the access to the Debian

Date: Tue, 26 Sep 2006 07:13:25 +0200

will "push" there. You can change that if

4. Push your changes:

$ mt $ branch
   -db=debian.mtn pull www.debian-france.org 'org.debian.*'

5. Create a working copy:

$ tar xzf
   {package} {version}.orig.tar.gz
$ mv {package} {version}.orig
   {package} {version}.orig
$ cd {package}
$ mt $ branch
   --db=debian.mtn checkout

6. Now you can build the package, change the

---

From: Pascal Obry <pascal@obry.net>
Subject: Re: ANN: Debian build scripts on
a public Monotone server
Newsgroups: comp.lang.ada

> Comparing Monotone to Ada to me is
high praise. If I understood you
correctly, you seem to be saying that
even though Monotone is only at
version 0.30, Monotone's "standards"
and professionalism make their v0.30
"better" than a lesser tools' 1.0+
version. Is that true?

Yes, it is true. Their emphasis on
correctness is truly astounding. It's the
attitude, man :)

> Comparing Monotone to Ada to me is
high praise. If I understood you
correctly, you seem to be saying that
Monotone is not perfect, and not 1.0
yet. The areas where perhaps it might be
weak is scalability and performance. In
my experience, it shines both for small
projects and for large projects with short
histories (few revisions). But, apparently,
as the number of revisions grows,
Monotone scales less than linearly.
They're working on it at the moment, with
the first notable scalability improvements
in the latest version, 0.30.

---

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Re: ANN: Debian build scripts on
a public Monotone server
Newsgroups: comp.lang.ada

I'd like to add that my confidence in
Monotone is not just because I like the
developers' attitude; it is also because:
- Monotone stores the SHA1 sum of
everything in the database, and verifies
the sum when checking out, updating,
merging, or syncing between databases.
Any data corruption will be detected,
and Monotone will crash rather than
corrupt data (never happened to me
though). (indeed, revision IDs are SHA1
sums, not "monotonic" numbers that
would make no sense in the context of a
distributed version control system).
- all commits to a database are signed by
a crypto key, and so authenticated in a
tamper-proof way.
- the database is an SQLite database; one
can always retrieve the data using SQL
commands, if things came to worst. In
fact, I have already, and successfully,
tampered with some trial databases using
SQL commands. That's the almost only
way one can delete data, BTW.

Now Monotone is not perfect, and not 1.0
yet. The areas where perhaps it might be
weak is scalability and performance. In
my experience, it shines both for small
projects and for large projects with short
histories (few revisions). But, apparently,
as the number of revisions grows,
Monotone scales less than linearly.
They're working on it at the moment, with
the first notable scalability improvements
in the latest version, 0.30.
What version did you test? Mercurial uses hard links when you work on the same file system, making cloning a very cheap operation. Moreover, since Mercurial 0.9 is out, the disk usage has been cut by 40%.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: [bug] the disk usage has been cut by 40%

I tried 0.8.1 and my test consumed 686 megabytes. If I take your word and reduce it by 40%, I get 411.6 Mb, which is still much, much more than Monotone's 183 megabytes (in Monotone 0.24) or 166 megabytes (in 0.26).

I stand by my opinion that Mercurial is only second best, after Monotone.

GNAT Split in Gentoo

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: "Split GNAT compilers" in Gentoo
Date: 2006-10-31 14:33:11

The advantage disappears as soon as you modify a file. Mercurial then breaks the hard link and duplicates the entire history for the file. In my trials with GCC, many files were changed between each version, so the space advantage was almost completely lost, even if the changes to each file were actually small.

The use of hard links is a poor kludge to minimise the impact of unshared histories.

> Moreover, since Mercurial 0.9 is out, the disk usage has been cut by 40%.

I tried 0.8.1 and my test consumed 686 megabytes. If I take your word and reduce it by 40%, I get 411.6 Mb, which is still much, much more than Monotone's 183 megabytes (in Monotone 0.24) or 166 megabytes (in 0.26).

I stand by my opinion that Mercurial is only second best, after Monotone.

The binary incompatibility is taken care of as described above, now on to the benefits.

The most obvious one (to me at least) is "following the Gentoo spirit"
— that is providing as much choice as practical.
You can quickly test various compilers (given general adherence to ARM of Ada packages it even may be possible to add non-gnat compilers to this mix) and how the libs behave when built with, say, gnat-gcc-4.1.1 vs gnat-gpl-3.4.6.2006. Now that the infrastructure is in place you can even easily create the ebuild for your own package and enjoy this automation.

OK, this is already quite long, so I'll stop here. You may also take a look at https://bugs.gentoo.org/show_bug.cgi?id=111340

this is the bug where most of the design discussion took place. Beware — it is quite a bit longer than even this description :).

Oh, one more thing (to avoid possible misconceptions). That long list of different gcc SLOTS I cited above does not mean that you can mix and match gcc version like the gnat ones — there is no Gentoo-wide setup like I just described for gnat (and it would not make sense system-wide). So, practically speaking with gcc you do have the flexibility of building your system with any of them, but you pretty much have to stick to a particular version. Although you *can* occasionally use some other version for a particular package.

---

References to Publications

AdaCore — GNAT Pro Insider

**RSS Feed: AdaCore Developer Center**

GNAT Pro Insider Nov 2006 issue available

Thursday November 30, 2006

The November 2006 issue of the GNAT Pro Insider newsletter is available for download. Contents:

* GNAT Pro and Ada 2005 Coming to .NET
* GNATstack Tool Available
* New GCC Technology
* What’s Coming in GNAT Pro 6.0.1
* New Version of gprmake
* New Version of GNAT Tracker
* Spotlighting a GAP Member
* AdaCore at Conferences
* Interview with Bob Dewar
* Public Courses at AdaCore New York
* New Target Platforms for GNAT Pro
* More Ada 2005 Features Available in GNAT Pro

To download the newsletter please [go to](http://www.adacore.com/home/company/press-center—su)

---

AdaCore — Ada 2005 for High-Integrity Real Time Systems

Ada 2005 for High-Integrity Real Time Systems (Video)

Monday September 25, 2006

An in-depth presentation by AdaCore senior software engineer, José F Ruiz, on Ada for embedded high-integrity real-time systems.

The talk covers:

* The Ravenscar tasking profile
* Flexible real-time scheduling algorithms
* CPU clocks and timers
* Timing events
* Flexible object-oriented features

Ada-Belgium — UML2 profile enforcing the Ravenscar Computational Model


2006-11-08

The slides of the technical presentation at the previous Ada-Belgium event are now available: "Correctness by construction: UML2 profile enforcing the Ravenscar Computational Model", June 2006 (Adobe Portable Document Format, pdf, 1520 KB). Updated the main page for the 2006 Ada-Belgium General Assembly + technical presentation and the full announcement of the technical presentation as well as the Ada-Belgium Meetings and Conferences page.

CrossTalk — AdaCore Quality Process

http://www.adacore.com/2006/10/31/ada-core-quality-process/

AdaCore Quality Process

Tuesday October 31, 2006

The Nov 2006 online issue of CrossTalk sees an interesting article on AdaCore’s quality assurance processes and tools:

“A software product is rarely a static artefact resulting from a one-time effort; it needs to evolve via periodic updates, to correct defects or meet new requirements, and it may need to be ported to multiple machines and/or operating systems. The development team might be distributed geographically, requiring careful coordination. A software producer must have well-defined processes for dealing with these issues, to ensure that its products successfully meet users’ needs.”


---

The Register — Mathematical Approaches to Managing Defects

September 20, 2006

SPARK is featured in the article "Mathematical Approaches to Managing Defects on The Register's developer site.

Embedded Systems Design — Programming Real-Time with Ada 2005

September 19, 2006


Safety-Critical Systems Symposium 2006


Tuesday October 24, 2006

Safety-Critical Systems Symposium 2006

AdaCore will present the following papers at this event:

* Ada 2005 for High-Integrity Systems By Jose F. Ruiz.

Embedded Systems Conference


Tuesday October 24, 2006

Embedded Systems Conference

AdaCore’s CEO, Dr. Robert B.K. Dewar will present a 90 minute class on: "Safety-Critical Design Techniques for Secure and Reliable Systems"

AdaCore will be exhibiting at DSO world (station #3 in the DSO Pavilion) April 6

Franco Gasperoni will be presenting the following paper at the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (taking place in parallel to ESC) April 4–7, 2006, Fairmont Hotel, San Jose, CA:

* Safety, Security and Object Oriented Programming

http://www.embedded.com/esc/sv/

Ada Inside

BAE Systems uses VectorCAST for testing


BAE Systems uses VectorCAST for testing

Mission critical Software
North Kingstown, RI—September 2006—Vector Software Inc., a leading provider of software test tools for embedded systems, today announced that BAE Systems has successfully used its VectorCAST product for testing safety critical software applications for the Hawk and Eurofighter aircraft.

BAE Systems has been using VectorCAST since July 2003, for module and SW-integration testing. The software tested with VectorCAST is primarily written in Ada and Ada 95, and runs on PowerPC targets.

According to Bill McCaffrey, Director of Sales for Vector Software, Inc., “We are extremely proud of our role as a technology supplier to BAE Systems. VectorCAST was specifically designed to support testing of complex avionics projects. Our robust integration with the Green Hills cross-development tools, and the uniquely position VectorCAST to support projects such as Hawk and Eurofighter.”

About Hawk
The Hawk aircraft, an advanced two-seat weapons systems trainer with enhanced ground attack capability. The aircraft provides fighter lead-in training and navigator and weapons systems operator training.

About Eurofighter Typhoon
Eurofighter Typhoon is the world’s most capable and dynamic swing-role combat aircraft. Developed by Germany, Italy, Spain and the UK, the Eurofighter Typhoon will fulfill European Air Force requirements well into the mid-21st Century. The aircraft is in full production and has been in service with all partner Air Forces since 2004. 638 aircraft are under contract for the four Nations and Austria, the first export customer.

About BAE Systems
BAE Systems is an international company engaged in the development, delivery, and support of advanced defence and aerospace systems in the air, on land, at sea, and in space. The company designs, manufactures, and supports military aircraft, surface ships, submarines, radar, avionics, communications, electronics, and guided weapon systems. It is a pioneer in technology with a heritage stretching back hundreds of years and is at the forefront of innovation, working to develop the next generation of intelligent defence systems.

About Vector Software
Vector Software, Inc. is a leading independent provider of automated test tools for software developers. Established in 1989 as a consulting and service organization, Vector’s product focus is to empower software professionals to deliver the highest quality software in the least amount of time. Vector’s "VectorCAST" line of products, reduce the burden placed on individual developers by automating and standardizing application component level testing. This innovative technology developed by Vector represents the “next generation” of intelligent embedded software test tools. The tools support Ada 83/95, C/C++ and Embedded C++ (EC++).

The market focus of Vector is on companies performing embedded systems development for aerospace, military, medical, telecom, and process control related projects.

Vector Software’s Product Family
- VectorCAST/Ada
- VectorCAST/C
- VectorCAST/RSP
- VectorCAST/Cover
- MC/DC add-on capabilities
- DO-178B Qualification Packages

Aonix Listed With Top Wind River Software Partners
http://www.aonix.com/pr_09.25.06a.html

Aonix Advances to Wind River Platform Partner Status
Aonix Listed With Top Wind River Software Partners
San Diego, CA, September 25, 2006

Aonix®, today announced a decision to strengthen its long-time partnership with Wind River Systems, Inc. (NASDAQ: WIND), the global leader in Device Software Optimization (DSO). Aonix is the provider of the PERC® real-time virtual machines and ObjectAda® real-time and safety-critical solutions for embedded targets. After years of serving as a Community Partner, Aonix will become a Wind River Platform Partner, the highest level of corporate partnership that Wind River extends to software companies. The Platform Partner level of participation is by invitation only, and requires Platform Partners’ technologies to be aligned with Wind River products from roadmap through development, QA, and test to ensure seamless integration.

Platform Partnership strategically positions Aonix with select upper-tier partners focused on driving joint business through one or more of Wind River’s vertical market segments. Aonix addresses Wind River’s Aerospace and Defence segment and offers years of servicing many customers with Ada real-time and DO-178B certified development solutions. Now, as a Platform Partner, Aonix is closely aligned with Wind River strategic direction for Java developers, providing industry-leading real-time virtual machines and tools tightly coupled with the Wind River development and execution environment.

“It is an honor to be selected by Wind River as a Platform Partner,” stated Gary Cato, Aonix manager of strategic alliances. “Plans are already underway for a joint safety-critical Java solution. Current Ada customers are enquiring about the potential of PERC, and C/C++ developers are eager for a better solution for their mission- and safety-critical applications as well. We look forward to expanding this interest to the larger Wind River audience.”

“Safety critical expertise and ability to address customers with large or complex projects sets Aonix apart from other vendors,” said Chip Downing, aerospace & defence industry marketing manager at Wind River. “Aonix’ development tools, especially the newer PERC real-time and safety-critical products, offer significant help to VxWorks platform users wanting to move up to a more modern, cost-effective development solution for ever-increasing levels of complexity.”

PERC Ultra is the ideal solution for embedded applications of high complexity, thanks to PERC Ultra’s predictable performance and its extensive support of off-the-shelf J2SE libraries and components. Its sister product, PERC Pico, meets the needs of resource-constrained hard real-time applications, featuring performance and footprint characteristics comparable to C. PERC Pico is smaller and faster than any other real-time virtual machine, yet it preserves key virtues of Java™ such as portability, reliability, and scalability.

PERC Ultra and PERC Pico are interoperable within a single application. For the first time, it is now possible for Java developers to create complete complex applications from infrastructure to the device level, without resorting to the use of other languages with less portability and robust memory use for specialized components.

Shipping and Availability
PERC Ultra for VxWorks 6.x is shipping now for Windows and Solaris hosts and PowerPC targets with AOT and JIT compilation. PERC development tools are available at no charge in combination with a maintenance contract. Target execution and deployment license pricing is based on projected volume. ObjectAda Real-Time for VxWorks is shipping now.

About ObjectAda®
ObjectAda provides an extremely effective solution for developing portable, highly reliable, and efficient applications. Based on Ada 95, the first and only internationally standardized object-oriented programming language (ANSI/ISO), ObjectAda is a truly multi-lingual environment. ObjectAda allows you to easily integrate Ada components with components written in Javabase, C, C++, FORTRAN, and other languages for...
Multi-lingual development. And ObjectAda works directly with commercial off-the-shelf libraries, components, and APIs.

For any platform, ObjectAda features a fast, open library model that is fully compatible with other languages, high-speed intelligent compilation, hyperlinked semantic browsers, syntax-directed editors, integration with configuration managers, and instant access to standard APIs. On all platforms, application generation is optimized for reliable, seamless execution of thread-aware applications within a safe and secure operational environment.

About PERC®
First introduced nine years ago, PERC is the most widely used real-time Virtual Machine available for Java developers, with fielded installations in telecommunications, telematics, avionics, deep space exploration, and office automation applications. PERC supports most major real-time operating systems and a variety of target processors including PowerPC, XScale, ARM, and Intel x86 architectures.

PERC Ultra is a virtual machine and toolset expressly created for demanding embedded and real-time systems requiring J2SE™ support. PERC Ultra delivers the ease and efficiency of Java™ Standard Edition support without sacrificing integrity, performance, or real-time behavior. It offers AOT and JIT compilation, remote debug support, deterministic garbage collection, standard graphics and extended commercial RTOS support. The PERC product line also features PERC Pico, a virtual machine designed for hard real-time applications requiring fast execution, small footprint, and access to low-level devices.

PERC Ultra is available for Wind River VxWorks 6.x. PERC Ultra is the industry-leading real-time virtual machine featuring the predictable performance critically needed for complex embedded applications. PERC Ultra offers the predictable performance and high levels of reliability fundamental to the military and aerospace markets where the VxWorks device software platform dominates. The high productivity required for these markets is supported by broad off-the-shelf Java™ component access made possible with PERC Ultra.

About Aonix
Aonix is a leading global supplier of technologies supporting the development of sophisticated applications primarily in the real-time and embedded domains. Our mission- and safety-critical solutions serve industries such as telecommunications, military and aerospace, and transportation. Aonix delivers the leading high-reliability, real-time embedded virtual machine solution for running Java™ programs deployed today and has the largest number of certified Ada applications at the highest level of criticality. Aonix also offers the TeleUSE line of Motif graphical user interface development solutions.

Headquartered in San Diego, CA and Paris, France, Aonix operates sales offices throughout North America and Europe in addition to offering a network of international distributors. For more information, visit www.aonix.com.

Indirect Information on Ada Usage

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

[...] Advanced programmer/analyst needed who will perform software development or software development support activities using the Ada programming language.

Needs to be an experienced user of UNIX and Ada, and have solid programming experience in large scale development.

Knowledge of air traffic control systems development a plus.

Candidate must have experience developing large systems and be able to work with a large team of developers.

Candidate must have knowledge of software development methodologies.

BS degree in Computer Science, IT, or other technical field required.

Within our aeronautics skills centre, we need Ada 83 or 95 software engineers specialised in Ada design and development.

Projects are varied and could reach managerial responsibilities depending on your experience.

You will be part of engineers team designing and developing new architectures, or working on others projects involving methodology and quality.

Experienced in Ada design, we propose you to join our team working on most challenging projects, in the highest speed area (space and avionics) and the railway sector. Knowledge in aeronautic or military standards or railway standards is a plus.

Profile:
You are Industrial Engineer (Ing) or Civil Engineer (IR) with good knowledge & experience in Ada 83/95 (min 3 years).

Very good communication skills and English speaking are mandatory as the development is done on a very international and multi-site basis, with frequent meetings and close interactions.

Besides a personalised career plan with real evolution prospects, we offer you an attractive treatment and various advantages which include a company car, a GSM and standard benefits such as luncheon vouchers, Group insurance, extra legal medical insurance, supplementary days off.

Several developers are needed to increase staffing levels on an air traffic control program in development (planned to go operational in the USA in 2009). The software is primarily written in Ada but also has interfaces with C++ components. The work consists of a mix of debugging and fixing problems, and new development. At least 6 months of Ada experience is essential.

From: jrig <jrig77@poczta.onet.pl>
Subject: Some Ada jobs statistics
Date: Sat, 04 Nov 2006 10:42:43 +0100
Newsgroups: comp.lang.ada

Some interesting Ada statistics gathered from job advertisements with “Ada” requirement.

http://www.itjobswatch.co.uk/contracts/uk/ada.do
http://www.itjobswatch.co.uk/jobs/uk/ada.do

Ada in Context

Consolidated Ada 2005 Standard

URL: http://adaic.com/whatsnew.html
November 10, 2006

The final version of the consolidated standards for Ada 2005 was posted. These documents combine the Ada 95 Standard, Technical Corrigendum 1, and Amendment 1.

When was Ada first standardized?

From: “Jeffrey R. Carter” <jrcaanter@acm.org>
Subject: Re: Basic Explanation of OO in Ada
Date: Tue, 19 Sep 2006 20:33:14 GMT
Newsgroups: comp.lang.ada

> C was invented 1973 and Ada 1983


From: Randy Brukardt <randy@rsoft.com>
Subject: Re: Basic Explanation of OO in Ada
Date: Wed, 20 Sep 2006 20:37:46 -0500
Newsgroups: comp.lang.ada

> Well that makes it 3 years for Ada to move from the it's beginnings to ISO-Standard — and 16 years for C.

The ISO standard for Ada was approved in 1987; it was the MIL-STD-1815A that was approved in 1983. While the content
was identical, it took 4 extra years for ISO approval.

About the Abs operator

From: Robert A Duff
Subject: Re: Why is abs an operator, not a function?
Date: Wed, 18 Oct 2006 21:45:24 -0400
Newsgroups: comp.lang.ada

> I'm a bit stuck trying to figure out how to coax more performance out of some Ada code. I suspect there is something simple (like compiler switches) but I'm missing it. As an example I'm using a simple matrix multiply and comparing it to similar code in Fortran. Since you are measuring real time and not CPU time, you might want also to take a bit larger matrices in order to have disk swaps and rests of initializations effects statistically small.

From: Gautier de Montmollin
Subject: Re: GNAT compiler switches and optimization
Date: Fri, 20 Oct 2006 14:12:26 +0200
Newsgroups: comp.lang.ada

One thing is already that you start the timer at the wrong place. You should start it after filling your array with random numbers. In the present state you compare the random generator, then your matrix multiplication. It is possible that Ada.Numerics.Float_Random.Range takes significantly more time due to Ada's quality requirements in the random generation.

On the other hand, switches you can try are:

- -O2 instead of -O3, -funroll-loops (usually good) -fstrict-math, for both Ada and Fortran -gnat, for Ada

Since you are measuring real time and not CPU time, you might want also to take a bit larger matrices in order to have disk swaps and rests of initializations effects statistically small.

From: Jeffrey Creem
Subject: Re: GNAT compiler switches and optimization
Date: Fri, 20 Oct 2006 11:56:50 -0400
Newsgroups: comp.lang.ada

I built the gcc "head" from gcc SVN with GNAT and Fortran to compare the same versions (at least as much as possible). I moved the start timing calls after the array allocation and filling so we just timing the matrix multiplication I moved the timing calls to make sure we were not timing IO in either case (both original versions were timing part of the "put").

Ada vs. Fortran performance

From: Duncan Sands <dhalbrick@free.fr>
Subject: Re: GNAT compiler switches and optimization
Date: Fri, 20 Oct 2006 13:42:23 +0200
Newsgroups: comp.lang.ada

> I'm a bit stuck trying to figure out how to coax more performance out of some Ada code. I suspect there is something simple (like compiler switches) but I'm missing it. As an example I'm using a simple matrix multiply and comparing it to similar code in Fortran. GNAT and Fortran to compare the same implementation usually special-case the overloading resolution for operator symbols, since they are so heavily overloaded. Maybe the fact that all predefined functions are operators simplifies that. Not a big deal, but it does have a certain uniformity — e.g. "not" is an operator symbol, too, and works the same way as "abs".

From: Martin Krischik
Subject: Re: GNAT compiler switches and optimization
Date: Fri, 20 Oct 2006 17:41:12 +0200
Newsgroups: comp.lang.ada

Indeed:

GNAT/GPL 2006 is based on gcc 3.4.6. For fortran you are using gcc 4.2.0. Try using comparable compiler versions, eg: an Ada aware gcc 4.2.0 (several linux distributions provide this) or a gcc 3.4.6 version of fortran (i.e. some version of g77).
I replaced the "random" data with some fixed same data just to be sure there was no funky "denormal" stuff happening that changed the speed.

Very little change in the order of magnitude that the original poster was seeing (I pretty much get results with GNAT running about 2.6 times slower) so it was time to look at the assembly.

I find it easier to read ass.embly using sse math so building Gnat via

gnatmake -g -f -gnatp -O3 -march=pentium4 -fomit-frame-pointer -mfpmath=sse tst_array.o

and Fortran via:

gfortran -O3 -g -march=pentium4 -fomit-frame-pointer -mfpmath=sse -c tst_array.f95

and then using

objdump -D -S tst_array.o

to look at them, you pretty quickly can see the problem.

The "inner loop" of the sse math [is 28 Instructions] vs. 8 for Fortran. [...] The GNAT version never stood a chance. It really seems like GNAT is dropping the ball here.

Granted small benchmarks can really lead one to believe things are better or worse than the truth but I don't think there is really an excuse in this case for this sort of performance.

From: Jeffrey Creem  
Date: Fri, 20 Oct 2006 19:52:54 -0400  
Subject: Re: GNAT compiler switches and optimization  

Newsgroups: comp.lang.ada  

Note, I am the first one to jump to the defence of "Ada" in general but in this case, GNAT just plain sucks compared to GNU Fortran as it does a poor job on (at least) the inner loop (verified by looking at the output assembly)

Jeff's (the other Jeff :) modified version looks a little cleaner and actually runs faster (than even old "fixed version" that did not time the IO and made sure to just time the matrix multiply in both versions) but it does not time the zeroing of the elements of C which would be required if this were a real matrix multiply routine and not some test driver.

However, even having said that, this not really equivalent version runs about 2x slower than the Fortran (with the same version of GCC)

I don't see any meaningful excuse for why GNAT should be slower in this case but it clearly is.

I tried looking at the tree dump generated by the front ends prior to going to the optimizer step (not something I have a lot of experience at) . One thing is clear is the trees generated by GNAT is quite a bit uglier and more verbose so it is not surprising that the optimizer is unable to fully clean things up resulting in the explosion of instructions at the assembly level. 

From: Jeffrey Creem  
Date: Sat, 21 Oct 2006 12:35:54 -0400  
Subject: Re: GNAT compiler switches and optimization  

Newsgroups: comp.lang.ada

> There is something strange. Martin Krischik was able to trim the overall time for the Ada code down to 24% of the first version (GNAT/GCC 4.1.1). This should make the Ada program as fast as the Fortran one, shouldn't it ? Maybe it's because the test is done on a 64 bit machine ? It needs some reconciliation. A good thing in that discussion would be that everybody shows each time: - which GCC version - which machine - the execution time of the multiplication for both Ada and Fortran - which version of the Ada code (matrix on stack/heap, Fortran or Ada convention)

I'd certainly be willing to run a few benchmarks but the important thing here is that rather innocent looking code is running 2-4x slower than it "should".

There are things that I think we can really rule out as being "the" factor. 1) Random number generator — I did timings (for both the Ada and Fortran) with timing moved to only cover matrix multiply.

2) Difference GCC versions — I built a fresh GCC from the GCC trunk for both Ada and Fortran

3) The Machine — I am running both on the same machine, though I suppose there could be differences in 32 bit v.s. 64 bit comparisons.

4) Runtime checks — both the original author (and I) ran with checks suppressed

5) O2/O3 — Actually, I could look at this some more with some other versions but a quick look when I first started seemed to indicate this was not the issue.

A few other thoughts.

Once the timing is limited to just the matrix multiply the stack/heap thing "should" generally not matter.

Some of the changes made to the Ada version make it not really the same program as the Fortran version and the same changes made to the Fortran one would also cause it to speed up (e.g. not counting the the zeroing of the target array during the accumulation phase).

I have certainly seen some amazing performance from some Ada compilers in the past and in general, on non-trivial benchmarks I am usually pretty happy with the output of GNAT as well but in this case it is not great.

Further, I tried playing a bit with the new auto-vectorization capability of the near 4.X series of GCC (has to be specifically enabled) and found that even very very trivial cases would refuse to vectorize under Ada (though after I submitted the bug report to GCC, I found that Fortran fails to vectorize these too).

One thing everyone needs to remember is that this example was (probably) not "Find the way to get the smallest value out of this test program" because there are always ways of doing some tweaks to a small enough region of code to make it better. If there is a 2-4x global slowdown in your 100KLOC program, you will never "get there" following the conventional wisdom of profiling and looking for the problems.

Now, I am not suggesting that GNAT is globally 2-4x slower than GFortran or anything like that (since that does not line up with what I have generally seen on larger code bases), but, if I were a manager picking a new language based on a set of long term goals for a project and saw that GNAT was running 2-4x slower and was still running 1.X to 3X slower after 2 days of Ada guru's looking at it, I'd probably jettison Ada (I know, I am mixing compilers and languages here, but in reality, that is what happens in the real world) and go with something else.

And before the chorus of "processors are so fast that performance does not matter as much as safety and correctness" crowd starts getting too loud, let me point out that there are still many segments of the industry where performance does still indeed matter. Especially when one is trading adding a second processor to an embedded box against a vague promise of "betterness" in terms of safety down the road. Ok …Off the soapbox.

So, in closing, if someone thinks they have "the best" version of that program they want timed against gfortran, post it here and I'll run them.

From: Jeffrey Creem  
Date: Sat, 21 Oct 2006 17:22:05 -0400  
Subject: Re: GNAT compiler switches and optimization  

Newsgroups: comp.lang.ada

> The first thing is to be sure that we are running the same program. Running your program with the following changes (as done in Fortran):

for I in A'range (1) loop  
for J in A'range (2) loop  

end loop;

for R in A'range (2) loop  

end loop;

I know these aren't the best versions, but I'm just trying to get the most out of this test program.
think Fortran float is a Long_Float, to be checked.
I went from 7.8s to 4.8s (with 1) and to 4.2s (with 2).

Actually, the original poster's Ada program had the temp var and all of my comparisons of programs that I have asserted were "the same" used the temporary.

As for Long_Float v.s. Short_Float, gfortran is using 32 bit floats (as verified by dumping the tree representation and assembly language).

Since we are all getting a bit confused by specific versions and numbers I thought I'd post a summary and create a way of tracking more global performance issues at the gnuaadafortran.com wiki.

The direct link to these test results are at the gnuada.sf.net wiki. tracking more global performance issues since we are all getting a bit confused by assembly language.

Actually, the original poster's Ada (what data is shared, multiple tasks doing "tricky" to really get things in parallel)

by dumping the tree representation and

gfortran is using 32 bit floats (as verified as a simple method of parallelizing code)

that's what got me started on this in the first place. I was introduced to OpenMP as a simple method of parallelizing code in fortran. Unfortunately it seems to be "tricky" to really get things in parallel (what data is shared, multiple tasks doing the same thing with the same memory, access violations, etc.) I remembered that Ada had tasking built in so I started playing with that. Now, as you can probably tell from my code, I haven't touched Ada in a very long time, but it was surprisingly easy set up a simple two-task test program.

Anyway, using the latest code from Jeffrey Creem, it looks like the execution time (on my machine) has been cut in half (9 seconds). The threaded version runs in nearly the same time for smaller problems but dies with a stack overflow for larger. I see a comment in the Bugzilla recommending a similar construct

```lang-ada
type Real_Matrix is array
   (1 .. N, 1 .. N)
   of Float;
```

That takes the memory from the stack rather than the heap though, no? I assume there is a compiler switch to increase the stack size so the code wouldn't die, but is that the "normal" way of allocating memory? I'm trying to not look like too much of an Ada neophyte :)

From: Gautier de Montmollin<gdemont@hotmail.com>
Date: Sun, 22 Oct 2006 14:31:05 +0200
Subject: Re: GNAT compiler switches and optimization Newsroups: comp.lang.ada

> Adding the Sum variable makes an important difference, as others have reported, in my case from 5.82 to 4.38s. Hoisting the indexing calculation for the result (C) matrix location is a basic optimization, and I would be surprised if it isn't done. The only thing I can think of is that it's a cache issue: that all 3 matrices can't be kept in cache at once. Perhaps compiler writers would be able to make sense of this.

The Sum variable was *removed* by someone at some point of the discussion in order to challenge a bit more the Ada compiler's optimizer. If you replace a good algorithm by a bad one, don't be surprised that the program is slow. At some point of bad coding the best code optimizer won't be able to help you. Eventually the optimizer will transform this:

```lang-ada
C (I, J) := Sum + A (I, R) * B (R, J);
end loop;
```

into something like:

```lang-ada
Sum := Sum + A (I, R) * B (R, J);
end loop;
```

Sum:=0.0;
for R in A'range (2) loop
Sum := Sum + A (I, R) * B (R, J);
end loop;
```

But still, it won't spare the time lost to fill the C matrix with zeros. If you want to do a benchmark with Fortran, it's really not a good idea to begin with "pessimizing" the Ada code.

From: Jeffrey R. Carter<jrcarter@acm.org>
Date: Sun, 22 Oct 2006 20:26:41 GMT
Subject: Re: GNAT compiler switches and optimization
Newsroups: comp.lang.ada

I did that in my 1st version. I wanted to see if the optimizer would result in equivalent code. No such luck. [...]

The initialization of C is static, so a good optimizer could. They're hard to find, though. [...]

I'm more interested in seeing what makes a difference in the Ada. In this case, the high-level features that let you write less code.

From: Tom Moran<tomoran@acm.org>
Date: Sun, 22 Oct 2006 13:01:53 -0500
Subject: Re: GNAT compiler switches and optimization
Newsroups: comp.lang.ada

Each 800x800 Float matrix is about 2.5 megabytes so I would expect two threads to be fighting each other over the cache. What are the single vs. dual threaded times for, say, 250x250 arrays and a 1 MB cache on your machine, and on the dual Xeon machine?

Curiously, on a 3 GHz Pentium 4 with 1 MB cache and compiling with -O2 -gnatp using the venerable gnat 3.15p, I get:

Time: 8.511855190 800 1 thread
Time: 5.682668332 800 2 threads
for a speedup of 33%
Time: 0.092724434 200 1 thread
Time: 0.059209520 200 2 threads
for a speedup of 36%

From: Jeffrey Creem<jjeff@thecreens.com>
Date: Sun, 22 Oct 2006 11:57:16 -0400
Subject: Re: GNAT compiler switches and optimization
Newsroups: comp.lang.ada

Followup on the bug report.

One of the comments asserted that the two programs were not equivalent though I am not yet 100% convinced that I believe it yet.

His recommendation was to remove a level of indirection by changing the way the array is declared.

N : Positive; := Positive'Value (Argument (1));
This does indeed substantially improve the performance (still not quite to Fortran levels). The reason I question the equivalence is that in the original version, the array could have been passed to a procedure that took in an unconstrained array (which is pretty much what I think the Fortran version would allow) while this new version would not do that.

A quick check shows the Fortran is now only 1.2 times faster (did not do the multiple run thing yet). Perhaps tonight. I'll also attempt to take one of the threaded ones and include that as well.

Can someone that understands Fortran better make an argument about the "closeness" of this approach vs. the other?

From: Jeffrey Creem
<jeff@thecreems.com>
Date: Mon, 23 Oct 2006 07:55:49 -0400
Subject: Re: Ada 2005 language update(s) & programmer productivity
Newsgroups: comp.lang.ada

> I doubt if anything will beat matmul (short of 640 000 processors). But Ada with explicit loops and Fortran with matmul are hardly equivalent. You need to convention-Fortran the Ada array type, import matmul, and call it to get a fair comparison. There shouldn't be much difference.

I looked at the gcc Fortran matmul. Unless there some additional trickery going on behind the scenes, it is not anything magical. It looks like a matmul implemented in C with manual array subscripting logic (i.e. uses a single dimensional array overlay).

In any case, it is not so much matmul I am trying to make faster here but rather just the nature of 2d array traversals in native language structures.

I just included the Fortran matmul to be "more than fair" to Fortran as I am in no way trying to bash Fortran.

The speedup for the tasks is quite odd though. I'll need to disassemble it tonight.

I also just finished a 4.0.2 install last night so I'll get those numbers to see if all of this mess is simply a regression someplace in the compiler.

From: Dr. Adrian Wrigley
<atmv@linuxchip.demon.co.uk>
Date: Sat, 21 Oct 2006 12:39:30 GMT
Subject: Re: GNAT compiler switches and optimization

Programmer productivity in Ada 2005

From: Martin Krischik
<krischik@users.sourceforge.net>
Subject: Re: Ada 2005 language update(s) & programmer productivity
Date: Sat, 04 Nov 2006 20:07:24 +0100
Newsgroups: comp.lang.ada

> What Ada 2005 language update(s) will turn out to have the biggest impact on programmer productivity?

Depends on the field in which the programmer works. My top three with a target desktop applications would be:
1) Interfaces
2) Container Classes Library
3) Anonymous Access Types

But if you do i.e. real-time work it will be something else. Ada is designed as a multi paradigm language from the beginning and as far as I know only Oz supports more paradigms as language feature (as opposed to library features). So in every domain programmers will choose their favourite new feature.

From: Randy Brukardt
<brukardt@rssoftware.com>
Subject: Re: Ada 2005 language update(s) & programmer productivity
Date: Mon, 6 Nov 2006 20:47:50 -0600
Newsgroups: comp.lang.ada

I agree with this; real-time programmers have very different requirements than those on desktops or even soft-real time servers.

I would probably say "limited with" (because it allows a natural structuring of interrelated packages that wasn't possible in Ada 95) and the containers library (because it makes something powerful available to everyone in a standard way).

But it's probably a little early to say (for instance, I haven't used any new Ada features in programs yet). The biggest value may be in something that appears less critical now.

About Ada Object-Orientation features

From: Jeffrey R. Carter
<jrcarter@acm.org>
Subject: Re: Basic Explanation of OO in Ada
Date: Mon, 18 Sep 2006 20:25:44 GMT
Newsgroups: comp.lang.ada

> I'm new to Ada, having to learn it for work, and I am beginning to understand the language, I think. However, coming from C (and family) and Java and other "modern" languages, I can't seem to wrap my head around Ada's OO methods. Is it that there is simply nothing like a class in C++ or Java?

You should realize that what you're asking about is not OO, but programming by extension. Because "OO" became a synonym for "good", and early examples of OO were shown in languages that supported (or required) programming by extension, programming by extension got called OOP, but that is a misnomer. Programming by extension does not necessarily have anything to do with object orientation. Programming by extension is an implementation technique.
Object orientation is a design attribute and may be implemented without using programming by extension.

From: Lucretia <lucretia@alycoos.co.uk>  
Subject: Re: Basic Explanation of OO in Ada  
Date: 18 Sep 2006 20:32:15 -0700  
Newsgroups: comp.lang.ada

This is very true and a lot of programmers don't realise they can develop in an OO way in asm or C (or any procedural language).

From: Jeffrey R. Carter  
<jrcart@acm.org>  
Subject: Re: Basic Explanation of OO in Ada  
Date: Tue, 19 Sep 2006 20:45:56 GMT  
Newsgroups: comp.lang.ada

OO is encapsulation of data and the operations on those data. In Ada 83, this was done through abstract state machines (package as object) or abstract data types ([limited] private type and corresponding operations). In assembler or C it's done through discipline.

Programming by extension is the use of type extension (type X is new Tagged_Type with ...); also called inheritance. Dispatching (also called polymorphism) can also be part of this, but in Ada, much of what would be dispatching calls in other languages are static.

Ada also has package extension (child packages) as a form of programming by extension.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>  
Subject: Re: Basic Explanation of OO in Ada  
Date: 19 Sep 2006 08:31:20 -0700  
Newsgroups: comp.lang.ada

> So basically, OO is the principles:  
* polymorph,  
* encaps,  
* inheritance.  
And Programming by extension is the way to do it in Ada, as class something:  
{}  
would be in C++?

Not exactly. That construct in C++ corresponds to several of Ada's constructs:

- a class in C++ is a unit of encapsulation, since it has public, private and protected parts. In Ada, packages, not types, are the units of encapsulation; they have public, body and private parts (respectively).
- a class T in C++ corresponds to a tagged type in Ada, but it also implicitly declares a pointer type (T*) and a reference type (T&) which correspond to Ada's class-wide access types; thus:
  class T {};  
is equivalent to  
package P is  
type T is tagged null record;

type Pointer_To_T is access all TClass;  
type Reference_To_T is access all TClass;  
end P;

Programming by extension does not necessarily involve inheritance. Consider:

package Pak2 is  
type T is private;  
-- not necessarily tagged procedure Proc (Object : in T);  
private  
type T is ...  
end Pak2;  
package Pak2.Extensions is  
procedure Additional_Operation (Object : in out T);  
type Extended is record  
Base : T;  
Additional_Data : ...;  
end record;  
end Pak2.Extensions;

This is extension by composition, as opposed to inheritance. Of course, you already understand how to program by extension by means of inheritance. There are still other ways, like "mix-ins", which combines composition and inheritance.

> One thing I'm still confused on is the differences between the type:  
A Vector is array (V : Vector);  
B Vector is array (V : in Vector);  
C Vector is array (V ; in Vector)  
D Vector is array (V : in out Vector);

D is the type where Vector is in out only.

I think about it this way. An object of a class-wide type is indefinite; we do not know its size, because its specific type may be anywhere in the inheritance tree. In contrast, an access-to-class-wide-object has a known size (usually just that of an address). As a consequence, you can create arrays of values, but not arrays of class-wide objects.

So, access types are not necessary to achieve polymorphism; the following achieves polymorphism without an access type:

package Pak is  
type T is tagged private;  
procedure P (Object : in out T);  
-- primitive operation  
end Pak;  
-- body omitted with Pak  
procedure Test (Object : in out Pak.TClass) is  
begin  
Pak.P (Object);  
-- dynamic dispatch  
-- achieves polymorphism  
end Test;

Access types come in handy if you want a collection of class-wide objects; the collection is said to be polymorphic with Pak:

package Collection is  
type Ref is access Pak.TClass;  
-- access-to-class-wide-objects  
type Vector is array (Positive range <>) of Ref;  
-- polymorphic vector  
procedure Traverse (V : in Vector);  
end Collection;  
package body Collection is  
procedure Traverse (V : in Vector) is  
begin

for J in V'Range loop  
if V (J) /= null then  
Pak.P (V (J).all);  
-- dispatches on the specific type of the object  
end if;
end loop;
end Traverse;
end Collection;

> Here's another interesting tip for C++ programmers. In C++, once you declare a member function to be "virtual", all calls to that member function dispatch dynamically, across the entire program.

Except when you ask it otherwise by qualifying the function name at the call site, see below.

> Conversely, if a member function lacks the "virtual" keyword, calls to it are always static. The only way you can determine whether calls dispatch statically or dynamically is by looking at the declaration of the member function. Things can become quite nasty if a class overrides an inherited, non-virtual member function, and makes it virtual.

Yes, it becomes nasty. It's a classical nono for C++ programmers.

> I'm not even sure what the language rules are in this case.

The static type used to call determines whether it dispatches or not, because it's the static type (of the pointer or reference) that allows to check whether the function is virtual.

> In contrast, in Ada, you do not declare primitive operations as "virtual", or "dynamic" or whatever. Instead, you choose *at the point of call* whether or not the call dispatches. Consider:

with Pak; use Pak;  
procedure Test (Object : in out TClass) is  
begin  
P (Object);  
-- dynamic dispatch  
-- achieves polymorphism  
end Test;

end Test;

Similar for C++:

p->fun(); // dynamic dispatch if in *p fun  
// is virtual  
p->BaseClass::fun(); // no dynamic  
// dispatch

> Consequences:

- you always know whether or not the call dispatches dynamically by just looking at the call
- you can choose how to dispatch based on your requirements
- a primitive operation cannot be "virtual" in some contexts and "non-
virtual" in others. It is always "potentially virtual".

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Basic Explanation of OO in Ada
Date: 19 Sep 2006 09:06:33 -0700
Newsgroups: comp.lang.ada

> Although you can only have vanilla arrays of definite types, you can have more sophisticated containers (for example Ada.Containers.Indefinite_Vectors) that store class-wide types in Ada 2005: package Pak_Vectors is new Ada.Containers.Indefinite_Vectors (Positive, Pak.TClass);

In this case, however, you store copies so some overhead can occur, but in many cases it can be more convenient than starting to play with pointers.

Doesn't Ada.Containers.Indefinite_Vectors have to declare a class-wide access type for its internal use? I'd guess so. But, it hides the access type from the user, and that's encapsulation at its best.

In Ada, one rarely needs access types at all. The only justification for declaring one is because you're doing some kind of dynamic data structure and allocating dynamically on the heap. With the standard Ada.Containers, you'd need access types even less often.

That's why in my examples I did not declare an access type along with the tagged type (in package P); instead, I declared an access type in another package (package Collection) because that package absolutely needed one.

Porting Ada code to Linux

From: Bill <william lugge@cisl.us Bài>
Subject: Re: Porting Ada code to Linux
Date: 31 Oct 2006 06:17:46 -0800
Newsgroups: comp.lang.ada

We have some code (about 60K lines) that was written for use on the Win95 platform as a collection of DLLs. We are now in the process of porting the parent application to Linux and would like to reuse the Ada code. The legacy code was written using the Aonix ObjectAda compiler/IDE and references packages like Win32 and Win32.Winfed.

We have been able to move the code over to the Linux side and compile using the GNU Ada compiler. However, to get this far we also had to copy the aforementioned "Win" packages over too. Clearly, this seems like the wrong thing to do, but it did get the compile to work.

The problem is that we are receiving a fair number of linker errors that seem to point to the file winbase.h.

So, it seems like there should be a Linux equivalent of Win32, but we have not been able to find it. Can anyone point us to a replacement for these packages that will make the compiler AND the linker happy? Is there anything else we should know about in our effort to complete this port?

From: Howard <Howard.Parrish@peterson.af.mil>
Subject: Re: Porting Ada code to Linux
Date: 31 Oct 2006 09:38:14 -0800
Newsgroups: comp.lang.ada

I work with Bill, and I wanted to provide a more detailed expansion of our situation;

We are porting from a Windows-95 "legacy" system to a new Linux one. The code we are re-writing is Visual Basic, and we are translating it into C++. But, the VB code had a dependency on a considerable amount of Ada 95 code that the VB code interfaced with via DLLs.

We need to port this Ada code over, also. We have decided, given that there is between 45 and 55 thousand lines of Ada code, with its attendant complexity, to simply use the Ada code as is without rewriting it into C++. This Ada code depends on some files, also written in Ada, that then have some pragmas to a Visual C++ library, "winbase.h"...

The Ada libs are;

- Win32
- Win32-Utils
- Win32-Winbase (pragmas to VCPP winbase.h.lib)
- Win32-Winfed
- Win32-Winnt
- Stdarg
- Stdarg-Impl
- Stdarg-Inst
- Stdarg-Machine

These are all Ada packages, and these name represent specs, and bodies, so named.

So, we are going to have to compile the Ada code, then link it with the new C++ code. We are using the GNU-G++ compiler linker, and we will need to be able to link in the Ada object files with our C++ files. We have to find a way to replace, or port, the mentioned Ada libs.

The "Winbase" dependency on the Visual C++ files is problematic, at best.

So, again, any insight that can be provided would be greatly appreciated.
You can write me directly.

From: Dmitriy A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Porting Ada code to Linux
Date: Wed, 1 Nov 2006 10:03:24 +0100
Newsgroups: comp.lang.ada

> They are not Ada libraries but Ada bindings. And the real problem you are going to have is to find a replacement for the Libraries those binding are binding to, i.e. for Win32* you will need a Linux replacement for "win32.dll".

I like to point out that you would have the same problem if the code was written in C (so it would be unfair to say "damn you Ada") in which case you would look for a replacement of "windows.h" and it's friend. It's low level operating system access and it just isn't there in Linux.

The bad news is: You need to replace it all with low level Linux access.

And also to be mentioned, the good news are: most of the code (the parts not bound to Win32) will be very easy to port, and that is because it is in Ada.

From: Steve <stevemd94@comcast.net>
Subject: Re: Porting Ada code to Linux
Date: Tue, 31 Oct 2006 19:28:41 -0800
Newsgroups: comp.lang.ada

My first response is: it is unfortunate that the Ada code was programmed with direct dependencies to these interface modules (apparently) scattered about the system. A better approach is to create a more generic binding to the system dependent interfaces to ease porting.

Given that you are talking about tens of thousands of lines of code, I would suggest attempting to create such an interface module with the existing code. Remove all references to the existing Win32 and Stdarg stuff. The compiler will help by telling you where things need to be "fixed up".

While this may sound like a daunting approach, in my experience this approach is not as bad as it sounds. It's a bit tedious, but really doesn't take that long.
For example eliminating the "With" statements will immediately identify things that need to be re-defined in an OS independent manner. The first compile may give you a LOT of errors, but often fixing one error fixes many. When a problem is identified, similar problems may often be corrected with a search and replace in an editor.

From: Dr. Adrian Wrigley  
<amtw@linuxchip.demon.co.uk>  
Subject: Re: Porting Ada code to Linux  
Date: Tue, 31 Oct 2006 20:21:36 GMT  
Newsgroups: comp.lang.adac

Rather than trying to convert the VB into C++, keep the Ada, convert the Ada into Linux and glue the C++ to the Ada, why don't you convert the VB into Ada?

You don't seem to have a strong technical reason to combine C++ and Ada, since C++ isn't a current implementation language. Because C++ and Ada have similar capabilities ("modern" compiled OO languages), the technical advantages of mixing them in new code are minimal. Mixing languages always has a cost – staff training, reduced compiler support, tricky interfaces, whatever.

Since you plan to convert to C++, you must have good reasons. Political? Skills sets? Anyone well skilled in both C++ and Ada would keep it all in the same language (Ada), or have a really good justification already for mixing (interfacing to existing, complex C++ libraries without Ada bindings, perhaps). But since you haven't chosen the Linux libraries yet, this can't be why.

I suspect you want to move to C++ for a mixture of political and skills reasons, but this really warrants planning to rewrite all* the Ada, in due course. History suggests this is an unsound approach when existing code works broadly as needed. If politics is absent, do it all in Ada, getting necessary skills from outside.

How to use anonymous access types

From: Pascal Obry <pascal@obry.net>  
Date: Sun, 19 Nov 2006 23:07:58 +0100  
Subject: Re: generic question  
Newsgroups: comp.lang.adac

There is no way to instantiate Unchecked_Conversion with an anonymous access type. This is something I found quite irritating in Ada 2005.

From: Pascal Obry <pascal@obry.net>  
Date: Tue, 21 Nov 2006 08:44:56 +0100  
Subject: Re: generic question  
Newsgroups: comp.lang.adac

Storage pools are associated with named access types. I think it's possible to allocate using an anonymous access type, but to me that's thoroughly confusing and so I have never attempted to do so.)

Yes of course:

C : access Integer;  
C := new Integer'(12);

>With anonymous access types the programmer must maintain the association between allocated instance and its pool. You can only allocate from a pool, and you must deallocate to the same pool from which you allocated, so at some point there needs to be a conversion between the anonymous access type and the named access type (with which the storage pool is associated).

I do not follow. The conversion will change nothing:

1. you allocate an anonymous access from a "compiler specific pool A"  
2. you convert the anonymous access type to a named access type : NAT  
3. you deallocate NAT from a "compiler specific pool B"

This will allocate and deallocable on different pool. And AFAIK there is no way to specify the anonymous access type pool.

I don't see a solution to that problem at the moment ... or I'm confused :

From: Pascal Obry <pascal@obry.net>  
Date: Tue, 21 Nov 2006 18:12:54 +0100  
Subject: Re: generic question  
Newsgroups: comp.lang.adac

 [...] You are saying that for every anonymous access type you need to declare a named access type from which allocate/deallocate. One point of the anonymous access type was to avoid proliferation of named access type.

I find anonymous access type less useful this way. Why even bother with them?

From: Randy Brukardt  
<randy@rrsoftware.com>  
Subject: Re: generic question  
Date: Wed, 22 Nov 2006 17:58:24 -0600  
Newsgroups: comp.lang.adac

> Because you can derive pointers without allocation.  
E.g. self references of limited types.

Well, that's one use. There are three others, IMHO:

* To stand in for the lack of "in out" parameters on functions;
* To avoid unnecessary type conversions between access-to-declared-in-limited-with;
* To stand in for the lack of subprogram types (anonymous access-to-subprogram parameters).

That's it. All other uses (especially controlling access parameters) are junk and should be avoided.

In part, that follows from my opinion that there should be no access types in the vast majority of reusable Ada packages (if the user needs dynamic allocation, they can provide it). Of course, the implementation of the such a package might have some access types, but those might as well be named (there is no need for implicit conversions there).

One of the reasons I feel this way that access types are inherently less safe than direct use of objects (because of the possibility of dangling pointers). And they force the customer of an abstraction to do memory management (or unsafe programming with very ugly calls) even when an stack-based allocation is fine.

The reason anonymous access parameters were invented was because some people wanted to copy dubious OO designs from C++ directly into Ada without appropriate conversion. Which just brings the flaws of those designs into Ada — yuck. And, yes, I was against the expansion of the uses of anonymous access types in Ada 2007. I lost that discussion primarily because I didn't have a reasonable alternative for the second bullet above.

From: Randy Brukardt  
<randy@rrsoftware.com>  
Subject: Re: Porting Ada code to Linux  
Date: Wed, 22 Nov 2006 18:02:55 -0600  
Newsgroups: comp.lang.adac

> The factory function (called by a user of the package to create instances of this specific type) should return an anonymous access type as its return type.

I don't think a factory function should ever return an access-to-object: it should return the object itself. If the client needs to allocate that dynamically, it can; else it...
can use the function for an appropriate static initialization.

It has been argued that sometimes you don't want those constructors to be inherited. In that case, it should return a classwide type, because even access-to-object is considered primitive and thus inherited.

But, of course, YMMV.

**User interfaces in AJAX**

From: Lucretia <lucretia9@lycos.co.uk>  
Subject: Should a GUI be separated from the application?  
Date: 26 Sep 2006 10:16:11 -0700  
Newsgroups: comp.lang.ada

When developing an application it is *mostly* combined with the GUI, a lot of toolkits do this. You would extend a UI tagged-type (or class, or whatever your language uses) and include the code inside the new tagged-type, an example would be a toolkit which provides event handler callbacks via primitive types (a la CLAW) thus forcing you to include your application code into the new derived type.

Is this the best thing to do? Should I give the users the option?

From: Tom Moran <tmoran@acm.org>  
Date: Wed, 27 Sep 2006 13:26:36 -0500  
Subject: Re: Should a GUI be separated from the application?  
Newsgroups: comp.lang.ada

> The users have the option of making the "application code" in the GUI framework just communicate with the real application, via rendezvous, sockets or whatever.

Agreed. It depends on the complexity and the timing requirements. If the response to a human-caused event can be done in a fraction of a second, the code might be included in the GUI event handler, while responses that take a long time should be separate to avoid blocking other events. If the complexity of communication is low — updating a percent complete display for example — then it's easy to separate app and GUI code and communicate with something simple like a protected object. If the communication is very complex, your user may find even rendezvous difficult and want to put the app code inside the event handler. One size doesn't fit all.

From: Simon Wright  
Subject: Re: Should a GUI be separated from the application?  
Date: Thu, 28 Sep 2006 07:09:19 +0100  
Newsgroups: comp.lang.ada

> What other design would you offer?  
What about having a browser as the GUI and using AJAX-style HTTP for the comms? The back-end would be something like AWS or my EWS (http://embed-web-srvr.sourceforge.net).

It does mean programming your GUI in HTML/JavaScript which I can agree is less than pleasant, but I'm not at all sure that GUI programming in Ada is any less tedious, and it certainly makes you think about separation of concerns!

I have yet to get to grips with XML-formatted (i.e., structured) responses. Perhaps I'm reading the wrong book (Professional AJAX from Wrox).

From: Pascal Obry <pascal@obry.net>  
Date: Thu, 28 Sep 2006 18:52:10 +0200  
Subject: Re: Should a GUI be separated from the application?  
Newsgroups: comp.lang.ada

Indeed, a very nice way to deal with GUI. This is what I call pluggable-GUI!

You can use AJAX-XML based control of the GUI. See the AWS's AJAX Web Elements demos. The application only output XML actions. No HTML or Javascript messing, the framework is already done by AWS.

From: Pascal Obry <pascal@obry.net>  
Date: Fri, 29 Sep 2006 19:54:18 +0200  
Subject: Re: Should a GUI be separated from the application?  
Newsgroups: comp.lang.ada

> I suppose that the necessary JS is created by AWS, then.

AWS comes with an AJAX runtime.  
> Doesn't that make it hard to use a 'standard' web designer to create the pages and just add a little interaction here & there?  
It must depend on the way you work. Usually that's not that hard.

From: Pascal Obry <pascal@obry.net>  
Date: Thu, 28 Sep 2006 20:46:50 +0200  
Subject: Re: Should a GUI be separated from the application?  
Newsgroups: comp.lang.ada

> Could you make a Pong game that way?  
Or a music synthesizer? Or does "GUI" mean a limited subset of graphical user interfaces?

I think with the current state of the technology a "limited subset" only. But in some years from now, we a nice SVG support... who knows! At least with AJAX the limited subset is far more advanced than it used to be.

**Are Ada 2005 containers thread safe?**

From: Maciej Sobczak  
Subject: Re: Multitasking and containers  
Date: Fri, 24 Nov 2006 09:51:24 +0100  
Newsgroups: comp.lang.ada

Paragraph 3 in Annex A says that it's OK to call any standard subprogram from concurrent tasks as long as the parameters do not overlap. John Barnes ("Programming in Ada 2005") suggests that in order to (for example) read from the same container, the operations need to be protected "by using the normal techniques such as protected objects". But reading from the protected object is not mutually exclusive (many readers are allowed) — so where's the gain? What's the difference between concurrent reads of, say, a Vector via a protected object vs. direct access?

From: Matthew Heaney  
Subject: Re: Multitasking and containers  
Date: Fri, 24 Nov 2006 12:02:31 GMT  
Newsgroups: comp.lang.ada

The reason is a conflict between safety and flexibility, a conflict that was resolved in favor of safety.

The container must set some internal state to indicate that Query_Element is executing, in order to prevent you from doing things inside Query_Element that would potentially destroy the element (such as Delete-ing it).

Even though Query_Element is technically a read-only operation, that's true only in the logical sense, not the physical sense. It doesn't look like Query_Element modifies the container, but it really does modify the container, to set some state that indicates a Query_Element is in progress.

Yes, it would seem as if it should be possible for multiple tasks to all be reading from the container simultaneously. But it's impossible to do that and also satisfy the requirement that the container detect potentially harmful manipulation of the container while Query_Element is executing.

So multiple tasks — even tasks only calling (logically) read-only operations — cannot simultaneously call container operations without also synchronizing the tasks, by wrapping the container inside a protected object, using a critical section, etc.

From: Matthew Heaney  
Date: Fri, 24 Nov 2006 12:13 GMT  
Newsgroups: comp.lang.ada

> That would be interesting, but would break apart when encapsulated within a protected object, because there multiple readers would be allowed.

But I think that's true only when multiple readers are calling protected functions. (There is a subtle difference in semantics between protected functions and protected procedures.) It does seem you'd need to use a protected procedure when manipulating a container nested inside a protected object, since a protected function wouldn't provide the level of synchronization required.
> Having a mutex for readers sounds like a concurrency killer and relying on protected wrappers seems to be fragile because of this possible mutability. So — what is The Solution (tm) for multiple tasks reading from the same container?

Declare the container object inside a protected object, and use protected procedures to manipulate the container. Protected wrappers should be fine, as long as you use protected procedures, not protected functions.

> Let's say you want to have N worker tasks consulting a shared dictionary (map) that was initialized before the tasks started their work. How would you solve this?

As above: declare the container object inside a protected object, and use protected procedures to manipulate the container.

From: Matthew Heaney  
<matthew@heaney@earthlink.net>  
Date: Fri, 24 Nov 2006 12:13:32 GMT  
Subject: Re: Multitasking and containers  
Newsgroups: comp.lang.ada  
> Write your own container. Parallel systems require delicate handmade work.

Horrible advice. Just declare the container inside a protected object, and manipulate the container by calling protected procedures. Works great...

From: Jeffrey R. Carter  
<jrcarter@on2.com>  
Subject: Re: Multitasking and containers  
Date: Mon, 27 Nov 2006 04:17:11 GMT  
Newsgroups: comp.lang.ada  
> I guess we have different interpretations of "delicate handmade work". I took it simply to mean custom implementations tailored to the specific project (such as a container that does not need protection for reads).

But that's the same as saying you cannot use anything in the predefined library. If so then the container library is the least of your problems!

From: Simon Wright  
<simon@pushface.org>  
Subject: Re: Multitasking and containers  
Date: Mon, 27 Nov 2006 21:15:29 +0000  
Newsgroups: comp.lang.ada  
If the predefined library doesn't meet your needs you do indeed have a lot of work ahead of you; often small beer in the overall scope (e.g., safety-related systems with SIL4 software => no runtime at all — GNAT high integrity edition for example — highly desirable if not absolutely necessary precondition for certification).

But if it does meet your needs you're crazy not to use it!

From: Matthew Heaney  
<mheaney@on2.com>  
Subject: Re: Multitasking and containers  
Date: 28 Nov 2006 09:12:27 -0800  
Newsgroups: comp.lang.ada  
> Using a protected object's procedure/entry would kill concurrency by serialization of the action to undertake.

There is a difference between "synchronizing access to a shared resource" and "waiting for a resource to become available".

Calling a protected function or procedure is an example of the former. Calling a protected procedure would hardly "kill concurrency". In a monitor there is only synchronization. (I think it's the case that the task stays in a running state.)

Calling a protected entry whose barrier condition is false is an example of the latter. If the barrier condition were false this would mean the task waits (it transitions to a blocked state). I would be loathe to say that that would "kill" concurrency since in typical designs that's exactly what the task is supposed to do.

From: Dmitry A. Kazakov  
<mailbox@dmitry-kazakov.de>  
Subject: Re: Multitasking and containers  
Date: Tue, 28 Nov 2006 19:21:42 +0100  
Newsgroups: comp.lang.ada

In absence of preemptive scheduling.

No difference if the above premise holds, i.e. no task switches as long as the barrier is closed.

But, the above is true *only* for a single-CPU system. So for a truly parallel system it could become a problem. Dual-cores aren't that expensive these days. [...]
The RM says that Inline is a recommendation that an implementation is free to ignore. Using an Inline that will get ignored will not make your program illegal according to the RM, but that is not* the same as saying that it shouldn't cause a compilation error; there are plenty of reasons why users might want a compiler to reject a legal program, and using an ignored Inline might be one reason, especially if a warning message might get missed during a "make". Maybe one of the GNAT flags Brian used told it to reject pragmas it couldn't handle. I don't know GNAT very well so I can't say.

Finally, the statement "The presence or absence of a pragma should not change a legal program to an illegal program nor illegal to legal" isn't true even for implementation-defined pragmas. 2.8(17) does say an implementation-defined pragma should not make an illegal program legal, but it also lists a couple exceptions to this rule. 2.8 also implies that a pragma that is not defined by the language nor by the implementation is ignored, but syntax rules still apply. However, if you use an implementation-defined pragma but violate the implementation-defined rules for that pragma, there's nothing I can see in 2.8 that says your program is still legal. That decision is left up to the implementation.

From: Robert A Duff  
<bobduff@shell01.TheWorld.com>  
Subject: Re: gnade error  
Date: Wed, 15 Nov 2006 08:29:26 -0500  
Newsgroups: comp.lang.ada  
> OK, a *legal* pragma should not ...  
> There are many pragmas that cause other parts of the program to be illegal. Pragma Restrictions, for example, has that as its main purpose. A pragma that goes the other direction (causes an otherwise illegal program to be legal) is in rather poor taste, but there are some cases of that, too — rather obscure cases.  
> But you're right about pragma Inline — a pragma Inline cannot affect the legality of the program (so long as the pragma itself obeys the rules, such as the names to denote subprograms).  
> However, many compilers have switches that invoke non-standard modes. In GNAT, you can tell it to warn about pragmas Inline that are not obeyed, and you can also tell it to treat warnings as errors (-gnatwe switch). In this non-standard mode, a pragma Inline might be one reason, especially if a warning message might get missed during a "make". Maybe one of the GNAT flags Brian used told it to reject pragmas it couldn't handle. I don't know GNAT very well so I can't say.

Managing large data structures

From: Alex R. Mosteio  
<alessandro@mosteio.com>  
Subject: Re: exception access violation  
Date: Wed, 15 Nov 2006 14:32:11 +0100  
Newsgroups: comp.lang.ada  
> Why is using the heap + controlled for larger data structures more portable than using the stack? I know that GNAT needs to be talked into providing sufficient space on the stack. You might be that running into this kind of stack trouble only when you port from another compiler to GNAT?  
I have no experiences out of GNAT, that be said first.  
In past times, I had to deal with implicit limits in Linux/ld (I seem to remember it was 2MB. This has changed with kernels and is no longer an issue). Because of this, porting from Windows to Linux was somewhat painful because problems not arising in Windows did arise in Linux (even using the usual linker stack options, so I had to dig for even more obscure switches). Other languages don't exploit so much the stack, so it is more difficult to find experiences in the area from other developers.  
Other point that may be relevant: by default, GNAT doesn't release memory historically claimed by stacks, contrarily to heap-allocated one. Couple this with lots of tasks or recursion and quickly memory can start to be an issue. (I'm not sure if this is Linux 2.6 default behavior or GNAT management of secondary stacks, couldn't locate it in the docs).
Finally is the trap I always fall for: "This is small enough to be in the stack". And then it grows, and then it starts to be a problem, or some recursive algorithm can't recur enough, and you end changing it or having humongous stacks or reworking algorithms.  
Of course, in real-time environments you'll prefer to know your memory needs from the start.  
> Should the decision whether some object lives on the heap or on the stack be based on compilers' support for dynamically sized local data structures?

As long as compilers are not perfect, I suppose it is at least an extra factor to consider. Also, I'd not limit it only to dynamically sized data.  
I've been bitten several times by stack-related problems when using GNAT, that's all I wanted to transmit (hence the 'in my experience' remark). If you don't have desires to learn about system internals, GNAT primary/secondary stacks and so... use the heap, Luke :)  
It is worrisome that I'm saying all this, because I find much more comfortable using the stack than any heap management.

From: Kevin K <kevink4@gmail.com>  
Subject: Re: exception access violation  
Date: Wed, 15 Nov 2006 23:43:01 GMT  
Newsgroups: comp.lang.ada  
Some operating systems put smaller than you would expect limitations on the environment stack. And while you can change it in some cases, while playing around, I found that Mac OS X, for example, seemed to have a 16MB limit (going by memory). And since task stacks were put on this stack instead of in the heap, that further limited it. If I was writing stuff from scratch that was also going to run on it, I would work on putting the data on the heap.

AdaCore — Internationalization in Ada 2005

Internationalization in Ada 2005  
Monday September 18, 2006

There are three aspects to making a language truly usable internationally.

First, which almost goes without saying, is that there should be an international standard that has been carefully reviewed by the international community. This is most certainly true of Ada, and specifically the draft standard for Ada 2005 has been approved by a vote of ISO member countries and is well on its way to getting the final stamp of approval. That vote was the critical one, it’s all smooth sailing from here and we may even have a formal standard before the end of the year. We were not sure if this would happen in 2006 or 2007 (which is one of the reasons we chose 2005 for the name). Of course once the standard is issued, the name of the language becomes simply Ada, since this name always refers to the current standard. It is notable that Java lacks such a standard, and has as a result been almost entirely a US-driven design.

Second, you want to be able to write programs that will handle foreign languages in a comprehensive manner. The standardization of international character sets has taken a huge stride forward in the last few years with the
The announcements of this kind in this forum so we would know who is still supporting Ada.


From: Jeffrey Creem jjeff@thecreems.com
Date: Sat, 07 Oct 2006 08:38:07 -0400
Subject: Re: ANN- AonixADT Ada Development Toolkit for Eclipse v. 3.11
Newsgroups: comp.lang.ada
I am starting to think that only us old timers ever know about Usesnet anymore.


Greenhills still has no Ada Eclipse but continues to update their Ada offerings (though their 2005 plans still seem in doubt). Though they tend to be aggressive in sales, you never see a new product announcement here.

Rational/IBM seems to have the problem that they are so big one can never be sure if they sell any tool anymore (not just Ada) since stuff is so bogged down in the "software is a service/craft you a solution" mentality.

Of course it could be that the vendors got tired of getting yelled at for commercial postings on Usenet. We are a hard group of people to please.

From: Marco <prenom_nomus@yahoo.com>
Subject: Ada vendors - Re: ANN- AonixADT Ada Development Toolkit for Eclipse v. 3.11
Date: 15 Oct 2006 10:27:38 -0700
Newsgroups: comp.lang.ada

Let's not hash Greenhills. They have provided good Ada support over the years. Ada 2005 updates should be customer driven, let's face it you can hardly buy a complete C99 compliant compiler 7 years later.

Rational/IBM still sells Ada but hasn't updated their products in years, unless you have to, I think it would be foolish to get Ada from them now.

From: Randy Brukardt randy@rrsoftware.com
Subject: Re: Ada vendors - Re: ANN- AonixADT Ada Development Toolkit for Eclipse v. 3.11
Date: Mon, 16 Oct 2006 20:10:03 -0500
Newsgroups: comp.lang.ada

I can't speak to their products, but several IBM/Rational people were instrumental in developing Ada 2005 — which is more than I can say about several other Ada compiler companies. I have to think that they are doing more than sitting on their hands between ARG meetings.

I do know that they don't have anyone turning out PR for the Ada group. (In IBM, it takes a lot of approvals to get PR allowed.) That might make updated products invisible.

From: Tom Grossman <grossman@aonix.fr>
Subject: Re: ANN- AonixADT for Eclipse now available for Intel/Linux and Sparc/Solaris for GNAT
Date: Tue, 31 Oct 2006 14:21:09 +0100
Organization: Aonix
Newsgroups: comp.lang.ada

> I'd like to say Thank You to those who made this plugin!
>
> It has feature that impress me. Many other plugins provide little more than syntax highlighting.

What could be an incentive to the business entity Aonix so they add all kinds of "free" to their product?

AonixADT when used with ObjectAda of course does not restrict project size. Mark, if you can show us a valid business model where we invest development resources to provide a free Eclipse interface for a competitor's product, we're willing to consider other approaches. We think that the limitation on project size means that the ADT is useful for smaller projects and especially in academic settings, without the Add-on. Judging from the number of students and faculty downloading the plugin, that may be the case. Because the downloadable version includes all features, we also think that users can get a good idea of whether it meets their professional development needs before requesting the Add-on.

> Will they have reason to think about conditions for projects that seem less self-interested and closed source, but rather seem good and profitable for everyone, including Aonix?

We are actively involved in projects like that. If you know of other such projects,
please let us know. We've been Ada advocates for 25 years. Still are.

From: Marc A. Criley <mc@mckae.com>
Organization: McKae Technologies
Subject: Re: [ANN]- AonixADT for Eclipse now available for Intel/Linux and Sparc/Solaris for GNAT
Date: Tue, 31 Oct 2006 19:30:35 -0600
Newsgroups: comp.lang.ada

> [Marc], if you can show us a valid business model where we invest
development resources to provide a free Eclipse interface for a competitor's
product, we're willing to consider other approaches. We think that the limitation
on project size means that the ADT is useful for smaller projects and
especially in academic settings, without the Add-on.

Perhaps not a "business model", but instead a "business opportunity model". All Ada software I've developed outside of my day job has been possible due to AdaCore's compilation tools, and the generosity of numerous other software developers in the Ada community. I've never used an Aonix product, but not because I have anything against the company (quite the contrary, I've been aware of, and appreciated, Aonix' Ada advocacy over the years).

I'm a serious, self-motivated Ada developer, as are many others in this forum, so any tool or product that I use has to be capable of performing effectively and efficiently in my development environment.

AonixADT for GNAT is too capability-limited for my activities, therefore I won't spend time giving a serious look to a tool that I can't properly evaluate in my real-world environment. So Aonix loses an opportunity to gain some mindshare and some hand's-on experience with me.

And without that mindshare and experience, there's no basis for me to recommend the product to others, including my day job employers. Fair or not, individuals tasked with recommending tools can't help but be biased towards those they're already familiar with.

At a former employer that was transitioning from a defunct Ada 83 compiler to Ada 95, it was my responsibility to make the technical case for the compiler choice—and clearly my 5 years of experience at that time with GNAT influenced my analysis. More recently, my experience with the free distribution of the Perforce Source Control tool (www.perforce.com) to small development organizations led me to recommending Perforce (over CVS and ClearCASE, which I also had significant experience with) to an employer that was upgrading from an ancient version of SourceSafe.

Hand's on exposure to fully capable tools, or those that are limited in a non-intrusive way (e.g., the free Perforce is limited to two users—not much of an impediment to the One Map Shop :-) gain mindshare for the product and the company. While _I_ may find it infeasible to drop $5000 for an Aonix Ada compiler on Linux, that's not to say I won't give it a fair hearing to an employer based on what I've learned about the company's products, quality, and service gained by the use of their other products—such as AonixADT.

**Ada and High-Integrity Systems**

From: Maciej Sobczak <maciej@msoxbczak.com>
Subject: Reference-oriented language and high-integrity software
Date: Fri, 03 Nov 2006 09:03:07 +0100
Newsgroups: comp.lang.ada

John Barnes in "Programming in Ada 2005", in the introductory section in the chapter devoted to access types, writes: "Java is currently popular. It has pointers which are called references. In fact almost everything is declared using references although this is hidden from the user. This means that Java is inappropriate for high integrity applications."

What is interesting is the following implication as John Barnes leaves without explanation:

references => no high integrity

It's also clear that the above statement applies not only to Java in particular, but to every other language that is similarly "reference-oriented". My question is: where this implication comes from?

Taking into account that JB also wrote a book about SPARK, some reasoning can be found there and my understanding (simplified) is that reference-oriented language implies a heavy use of dynamic memory, which makes it impractical/impossible to perform any static analysis of memory consumption. Garbage collectors add their own factors to the problem.

Is the above a reasonable explanation? Is it the only one? What else makes the reference-oriented languages inappropriate for high-integrity software?

And last but not least, how does the JB's statement stand in front of things like Real-Time Java or even HIJA (High-Integrity Java)?

From: Rod Chapman <roderick.chapman@gmail.com>
Subject: Re: Reference-oriented language and high-integrity software
Date: 3 Nov 2006 01:43:18 -0800
Newsgroups: comp.lang.ada

The provision of sound (i.e. no false negatives) and fast aliasing analysis a key factor, even in the absence of dynamic memory and garbage collection.

The soundness (and efficiency) of the information flow analyser and the VC Generator (which is basically an implementation of Hoare's assignment axiom) depend on this property.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Reference-oriented language and high-integrity software
Date: Fri, 03 Nov 2006 12:15:44 +0100
Newsgroups: comp.lang.ada

The other part of the explanation, AFAIU, is that a reference can go wrong, i.e. point to deallocated memory, to unallocated memory, or to the wrong piece of memory. References also introduce aliasing, i.e. two references can point to the same item. All these make it almost impossible to statically prove that no unintended side effects ever occur in the program (correctness means: do what you're supposed to do; safety means: do not do what you're not supposed to do. It is this latter part that matters to the present discussion).

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Reference-oriented language and high-integrity software
Date: Fri, 03 Nov 2006 15:27:56 GMT
Newsgroups: comp.lang.ada

And your rebuttal can be rebutted at the highest criticality levels where you do not certify the source text, but the object code emitted by the compiler. In those contexts you do not even trust the compiler. References make the object code even more difficult to certify.

> This makes sense in case of Java, but one could also argued that immutability of objects — a common feature in some reference-oriented languages — can make it less severe.

Yes, provided you trust the compiler — which you don't in high-integrity software.

> So — let's imagine a language, which is reference-oriented with all objects immutable. Apart from dynamic memory, is there any problem?

Yes. Tracing the object code to the source text, and certifying the object code. I'm not saying it's impossible to do; just that it's unacceptably expensive to do.

From: Dr. Adrian Wrigley <awmtv@linuxchip.demon.co.uk>
Subject: Re: Reference-oriented language and high-integrity software
Date: Fri, 03 Nov 2006 15:27:56 GMT
Newsgroups: comp.lang.ada
Newsgroups: comp.lang.ada
> Neither Real-Time Java nor HIJA can reasonably be described as Java. The last time I looked, both required special compilers; were designed to produce native machine code not an interpreted J code; allocated objects on the stack rather than heap; and had no garbage collection. The restrictions mean that you cannot use any of the standard libraries and don't get platform portable code; the two primary attraction of Java in the first place. What you do have is two new languages that just happen to have a Java-like syntax. The modifications required to give these new languages their real-time and high integrity credentials are precisely aligned with John Barnes's objections to them.

So the advantage is that you can attempt use existing programmers for writing high integrity/real-time software. And the code produced can even be executed with standard compilers/runtime, albeit without the benefits of the HI/RT environment. And of course, all the IDEs, code analysis tools can be used. Sounds rather useful.

The sceptics are saying the only benefit is better buzz-words. But then SPARK Ada is based on the same philosophy (restricted language, compiler, run-time to achieve tougher HI/RT goals).

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: Re: Reference-oriented language and high-integrity software
Date: Fri, 03 Nov 2006 18:30:56 +0100
Organization: Adalog
Newsgroups: comp.lang.ada

Experience shows that people overestimate the time to learn a new language, and under-estimate the time to train people to the constraints of writing high integrity/real-time software. Better take an experienced real-time programmer and teach them Ada, than the other way round!

Costs of Commercial Open Source Software

From: Larry Kilgallen <kilgallen@SpamCop.net>
Subject: Re: Why is OSS Commercial Software So Expensive?
Date: 12 Oct 2006 05:07:36 -0500
Newsgroups: comp.lang.ada

> well, but what if I want just a compiler without any support and just for 3 seats... not for 5, and not with GPL?

The president of AdaCore has said that below a certain level it is more expensive for them to support a smaller number of users than a larger number. When one gets to 5 users, there is some self-help provided between people in the customer organization.

From: Pascal Obry <pascal@obry.net>
Date: Thu, 12 Oct 2006 20:28:17 +0200
Subject: Re: Why is OSS Commercial Software So Expensive?
Newsgroups: comp.lang.ada

> Well, not only, AdaCore also sells a non-GPL version of the compiler with the guarantee that it is ok to use it in a proprietary context.

Isn't this the case with GNAT/FSF too?

We already had this debate here, and I certainly do not want to restart it. But we have 3 GNAT compilers: GNAT Pro, GNAT GPL, GNAT/FSF. I think that every needs are covered. Not talking about other vendor's offerings. I find it hard to believe that some people can't find the right tool for their work.

From: Pascal Obry <pascal@obry.net>
Date: Fri, 13 Oct 2006 18:36:11 +0200
Subject: Re: Why is OSS Commercial Software So Expensive?
Newsgroups: comp.lang.ada

> The FSF doesn't give you a "guarantee* (in the form of a signed document). If someone stands up and says "you cannot use this part of GNAT in a proprietary context because I have a copyright on this part and didn't allow it to be used in such context", AdaCore would handle the problem and deal with the claim (if you have the signed guarantee), while you have no such guarantee with the FSF.

Ok, ok and millions of people are developing software with C/C++ using GCC/FSF without trouble :)
GNAT is GMGPL or GPL; there is no license to buy. The only thing you can buy is support.

If you don't need support, just use a publicly released version of GNAT, either from https://libre2.adacore.com/, or from an FSF GCC distribution.

Hmm. The latest public GNAT from https://libre2.adacore.com/ is under the GPL license. Perhaps you mean you want GMGPL instead of GPL; is that the issue?

In that case, you can use the FSF GCC distribution of GNAT; it is GMGPL. However, it is pretty broken, compared to the https://libre2.adacore.com/ distribution. But that just means you actually do need support, and we are back to option 1 :) .

From: Stephen Leake
<stephen_leake@stephe-leake.org>
Date: Thu, 12 Oct 2006 14:39:46 -0400
Subject: Re: Why is OSS Commercial Software So Expensive?
Newsgroups: comp.lang.ada
> In my opinion AdaCore is not asking too much for commercial support.

I agree whole-heartedly; the support they provide is well worth it. I wish the other companies I deal with had an option to pay more for better support. Instead, I have to spend my time hassling them to let me talk to the people who actually know something, instead of the front-end support filter people. It ends up costing me more than AdaCore does.

> Being a small company, of course, will have to consider spending so much that is over budget!

I do have the luxury of working for a big company; NASA. Sometimes, we manage to do things right :) .

The Ada mindset

From: Jeffrey R. Carter
<jrcarter@acm.org>
Subject: Re: basic basic ada question
Date: Fri, 20 Oct 2006 03:10:05 GMT
Newsgroups: comp.lang.ada
> Dijkstra wrote an interesting paper called "Why numbering should start at zero", which you can find via Google. I don't buy it — I like to number most things starting at 1, despite his fairly reasonable arguments to the contrary.

I've seen it, and I don't buy it, either. My experience is that fewer mistakes are made when the numbering from the domain is used. Some of his arguments, such as ease of calculating the length of a sequence, are things that the language should do for you. There should be Length for discrete subtypes; it returns the number of values in the subtype.

> It's interesting that for enumeration types, T'Pos starts numbering at 0.

Yes. They're not Pos-itions, they're offsets. Positions should start at 1. If they did, then T'Base'First could be 0. That might be useful in some cases.

> Why should one think in C or Ada?

There are lots of concepts that transcend the languages. After all, both languages have subroutines, parameter passing, stack/heap allocation, etc, etc.

The Ada mindset is essentially the same as the SW engineering mindset.

From: Maciej Sobczak
<maciej@msobczak.com>
Subject: Re: basic basic ada question
Date: Fri, 20 Oct 2006 09:13:00 +0200
Newsgroups: comp.lang.ada
Cool sentence, but I know another: "Computers think in C", Both are good (and actually used) in flame-wars.

From: Jeffrey R. Carter
<jrcarter@acm.org>
Subject: Re: basic basic ada question
Date: Fri, 20 Oct 2006 20:39:55 GMT
Newsgroups: comp.lang.ada
Computers don't think, but they follow instructions in object code. "Real men write object code."

It indicates a good point: SW engineers deal with abstraction and try to ignore or hide implementation details as much as possible. Coders like to try to do everything at a low level.

From: Maciej Sobczak
<maciej@msobczak.com>
Subject: Re: basic basic ada question
Date: Fri, 20 Oct 2006 09:25:44 +0200
Newsgroups: comp.lang.ada
Most of what you know from C/++ is wrong for Ada.

It's a bit of overstatement and I, personally, don't find it to be the case. I would rather claim that if someone brings some bad habits from C or C++ to Ada, they were already bad habits in C and C++ anyway.

> The sooner you can stop thinking in C++ and start thinking in Ada, the easier you will find it.

Thinking in Ada is probably the same bad idea as thinking in C or C++ — at least if the final goal is not to have fun with the language (that's also a valid reason to write programs, really), but rather to build good final software. In this latter case thinking in terms of good engineering principles is the keyword — and then either any given language makes the implementation of these principles possible or not. This means that "thinking" in any particular language is already a bad idea and *using* the language to implement the chosen engineering principles is much more correct.

From: Jeffrey R. Carter
<jrcarter@acm.org>
Subject: Re: basic basic ada question
Date: Fri, 20 Oct 2006 20:54:51 GMT
Newsgroups: comp.lang.ada
Overstatement tends to attract the attention, and thus get the reader to think about it. Saying "Ada is different from C" is obvious, and will be ignored. You'll still get Ada-C, with pointers everywhere. Saying "Everything you know from C is wrong" might get him looking for ways to do things differently.

I should have said "the Ada mindset", which is also known as "the SW engineering mindset", but when you're knocking out these posts you often don't have time to perfect your wording. But...

Thinking in terms of the language is not as good as thinking in SW engineering concepts, but when a person is thinking in C/++, thinking in Ada is much closer to those concepts than the way the person is thinking. Getting a coder to consider thinking in such concepts is harder than getting him to consider thinking in another language. Once he starts thinking in Ada, the final step is much easier.

Event mechanisms for GUIs

From: Lucretia <lucretia9@lycos.co.uk>
Subject: Event mechanisms for GUI's
Date: 26 Sep 2006 10:28:04 -0700
Newsgroups: comp.lang.ada
My wxAda project stalled due to a problem with C++. I'm starting on a new project to see if I can do better than wxAda using native Ada and binding only where necessary. It will do what wxWidgets does and provide native controls for different platforms, i.e. GTK+-2.x for Linux, Win32 API, etc. I'll be implementing it in Ada 2005 as well, as long as the FSF GNAT doesn't cause too many problems.

I'm currently thinking of the event handling mechanism, I started to think about using the listener/subject pattern, but then started to wonder about the alternatives:

1) Event loops
   This would be painful as there'd be a lot of case statements.
2) Signals & slots
   This just seems to be a simplification of (5), where instead of using an observer interface you just pass a function pointer and the signal can contain many of these function pointers.
   Can be implemented using generics to provide type safety.
3) Extension of tagged-types (like CLAW)
   Not a bad idea, just override the subprograms you want to know about.
4) Ada.Real_Time interrupts
I don't know enough about this, so I won't even comment.

5) Listeners/Subjects
I'm not too sure how Java handles this with their Listener interfaces, I can only think of having a listener/subject pair per event type and then having something similar to the listener interfaces that Java has by calling specific procedures in the interface/class from the listener's update procedure.

So, there would possibly be 3 types: observer (abstract tagged), subject (tagged), listener (interface). Then the control (e.g. a window) would then have to implement an "Add <whatever>_Listener" per event type, which would add a <whatever>_Listener to a <whatever>_Observer. Seems overly complex to me.

The one thing that Ada would have over the Java implementation is the use of null subprograms in the interface, therefore an adapter type would not be necessary as you would be able to implement only the subprograms you are interested in.

From: Jeffrey R. Carter <jrcarrier@acm.org>
Subject: Re: Event mechanisms for GUI's
Date: Tue, 26 Sep 2006 20:39:27 GMT

The best way is to have a protected event queue for each window. There should be mechanisms to specify what events are and are not put on the queue, and for combining the queues for multiple windows into one queue.

There's nothing wrong with case statements. They're very clear and easy to read and understand. Remember Ada's explicit design goal: "[1]Enhphasis was placed on program readability over ease of writing." [ARM Introduction] If you have a problem with writing case statements, then you probably haven't adopted The Ada Way yet.

From: Tom Moran <tmoran@acm.org>
Subject: Re: Event mechanisms for GUI's
Date: Tue, 26 Sep 2006 18:18:10 -0500

There's nothing wrong with case statements. They're very clear and easy in small quantity, yes. But many-branched, nested, case statements with a lot of near duplication are not so clear or maintainable. They are a very low abstraction level, computer-centric, device.

If This_Window is for showing danger warnings in big red letters to the reactor operator, and That_Window is for showing daily power demand over the last month, a mouse click in either is similar from a computer-centric view, but vastly different in the problem space. Having This_Window_Type objects and That_Window_Type objects as descendents of Basic_Window_Type (inheriting what they have in common) makes them different objects, and a dispatching call on When_Left_Button_Down(The_Clicked_Window) is in effect a succinct abbreviation for a case statement on the type of Window, implemented efficiently and without forgetfulness or typing errors, automatically by the compiler.

From: Randy Brukardt <randy@rssoftware.com>
Subject: Re: Event mechanisms for GUI's
Date: Wed, 27 Sep 2006 18:05:22 -0500

First of all, there is no simple or even satisfactory solution to this problem. The trouble is that handling asynchronous events in a sequential programming language (or even a high-level parallel one like Ada) is a poor match. You complained about "a subprogram being called when a framework pleases", but you have the same effect with any mechanism — the order of operations depends on the events, not on the program. And any well-designed framework will make it clear when subprograms will be called and by which task.

Second, explicit event handling is among the worst solutions, because it requires an explicit response to every possible event. A real GUI system will have dozens, if not hundreds, of possible events for each Window. It's easy to forget to handle some of those events; moreover any *explicit* code necessary to do the *default* action reduces the readability of the system. And the easiest solution (the others clause) completely eliminates the main advantage of using case statements — the completeness check.

I'm not convinced that the "subprogram extension" solution used by Claws is the best one, and I'm certainly not convinced that the multi-task arrangement used by Claws is a good idea. (For a lot of systems, a single task solution with an explicit call-me-now routine would be easier to work with. But it would also make it a lot easier to starve the GUI — a problem that explicit events certainly have as well.)

My final conclusion is that all of the solutions have severe trade-offs; none is anywhere near optimal. My advice to the OP is to select a strategy that works well for the problems that they want to solve, and not worry too much about "goodness" — it's not going to happen in a GUI (that's why GUI builders are so indispensable).

Building a Library in Ada
From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: GNAT, shared libraries, building in different directories...madness!

I'm not convinced that the "subprogram extension" solution used by Claws is the best one, and I'm certainly not convinced that the multi-task arrangement used by Claws is a good idea. (For a lot of systems, a single task solution with an explicit call-me-now routine would be easier to work with. But it would also make it a lot easier to starve the GUI — a problem that explicit events certainly have as well.)

My final conclusion is that all of the solutions have severe trade-offs; none is anywhere near optimal. My advice to the OP is to select a strategy that works well for the problems that they want to solve, and not worry too much about "goodness" — it's not going to happen in a GUI (that's why GUI builders are so indispensable).

Building a Library in Ada
From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: GNAT, shared libraries, building in different directories...madness!
"build" & Compiler & "/" & Target & "/" & Build; -- no "temp"
-- No package Compiler, since we --> won't compile the library. But --> we can make it convenient for --> users to link with the shared library.
Linker_Switches := "-ladael";
package Linker is
for Default_Switches ("Ada") use
(Linker_Switches);
end Linker;
end Adael;

The third project file would build the samples. You would encourage your users to write similar project files for their own programs.

with "adael";
project Adael_Samples is
  Compiler := External ("COMPILER", "gnat");
  Target := External ("TARGET", "Linux");
  Build := External ("BUILD", "release");
for Source_Dirs use
  ("src/samples/**");
for Object_Dirs use
  ("src/objects/**");
for Executable_Dirs use
  ("build/Ada\_Target\_Samples\_Dir");
for Executable_Dirs use
  ("build/Ada\_Target\_Samples\_Dir");
package Linker is
  Linker_Switches := ":-ladael"
end Linker;
end Adael; Samples;

Now, the Makefile would glue them all together, create directories as needed, and simply call gnatmake:

# O mighty Emacs, this is a * Makefile * ...
.SUFFIXES:

# You can override these from the # command line:
COMPILER := gnat
TARGET := Linux
BUILD := debug

# Nothing should change past this point
lib dir :=
build/$(COMPILER)/$TARGET/$(BUILD)
lib build dir :=
build/$(COMPILER)/$TARGET/temp/$ (BUILD)

for Source_Dirs use
  ("src/samples/**");
all: $(lib build dir)/libadael.so
$(lib build dir)/test_app

clean:
  rm -rf build

args := -XCOMPILER=$(COMPILER) -XTARGET=$(TARGET) -XBUILD=$(BUILD)
$ (lib dir)/libadael.so: build_adael.gpr |
$ (lib dir)/lib_dir $(lib build dir)
gnatmake -c -Pbuild_adael.gpr

$(args)
gnatcc -shared -o $@

$(args)
$(lib build_dir)/*.o

-Wl,-soname,$(notdir $@) -lgnat

$(lib_dir)/chmod a=r $(lib dir)/*.ali

if [ $(release,$(BUILD))]
  strip $@
endif
$(lib dir)/test_app: adael_samples.gpr adael.gpr
$(lib dir)/test_app: $(lib dir)/libadael.so |
$(samples build_dir) -gnatmake -Padael_samples.gpr
$(notdir $@) $(args)

if [ $(release,$(BUILD))]
  strip $@
endif

$(lib dir) $(lib build_dir)
$(samples build_dir): 
  -mkdir -p $@
  -Phony: all clean

I think you *could* do away with GNAT project files, but that way lies madness.

From: Simon Wright
<simon@pushface.org>

Subject: Re: GNAT, shared libraries, building in different directories...madness!

Date: Sun, 15 Oct 2006 12:00:43 +0100

Newsgroups: comp.lang.ada

> There is a major problem with this source directory directive. The generic directory should contain all source for a GUI toolkit neutral UI, in a way that wxWidgets does currently.

We build several executables, each with its own set of supporting application-, intermediate- and device-level packages plus a fair bunch of common stuff. We have a main GPR for each executable (which can be extended by anyone who needs to build test programs), which calls up a machine-generated GPR (one for the whole project) which has no associated source code but instead defines lots of names for various sets of paths. These names are what are used by the main GPRs.

Each executable can be built in various contexts; variant A or B, runs on the host or on the target, etc. This is managed using case statements in the machine-generated GPR.

```makefile
Host_Network_Path := 
("a/b/c", "a/b/d");
Target_Network_Path := 
("w/x/y", "w/x/z");
case Platform is
  when "host" =>
    Network Path := Host_Network Path;
  when "target" =>
    Network Path := Target_Network Path;
end case;
```

 Maybe I should have explained that better in my first reply.

Now, since you seem to be running on an operating system that has no policy as regards library placement, maybe it is necessary to specify a library path at some point. But the OP did mention Linux, and to me this implies the GNU linker and tool-chain, and some sort of conformance to the file system Hierarchy Standard.

Come to think of it, it really better not to write the linker path in the project file, because the linker path tends to be system-dependent, whereas the project file is platform-neutral. I prefer to provide the necessary "glue" in the Makefile.

From: Simon Wright
<simon@pushface.org>

Subject: Re: GNAT, shared libraries, building in different directories...madness!

Date: Sun, 01 Oct 2006 18:28:18 +0200

Newsgroups: comp.lang.ada

> I think you *could* do away with GNAT project files, but that way lies madness.

From: Ludovic Brenta <ludovic-brenta.org>

Subject: Re: GNAT, shared libraries, building in different directories...madness!

Date: Sun, 01 Oct 2006 18:28:18 +0200

Newsgroups: comp.lang.ada

> I think you *could* do away with GNAT project files, but that way lies madness.

From: Simon Wright
<simon@pushface.org>

Subject: Re: GNAT, shared libraries, building in different directories...madness!

Date: Sun, 01 Oct 2006 18:28:18 +0200

Newsgroups: comp.lang.ada

> I think you *could* do away with GNAT project files, but that way lies madness.

From: Simon Wright
<simon@pushface.org>

Subject: Re: GNAT, shared libraries, building in different directories...madness!

Date: Sun, 01 Oct 2006 18:28:18 +0200

Newsgroups: comp.lang.ada

> I think you *could* do away with GNAT project files, but that way lies madness.

From: Simon Wright
<simon@pushface.org>

Subject: Re: GNAT, shared libraries, building in different directories...madness!

Date: Sun, 01 Oct 2006 18:28:18 +0200

Newsgroups: comp.lang.ada

> I think you *could* do away with GNAT project files, but that way lies madness.

From: Simon Wright
<simon@pushface.org>

Subject: Re: GNAT, shared libraries, building in different directories...madness!

Date: Sun, 01 Oct 2006 18:28:18 +0200

Newsgroups: comp.lang.ada

> I think you *could* do away with GNAT project files, but that way lies madness.

From: Simon Wright
<simon@pushface.org>

Subject: Re: GNAT, shared libraries, building in different directories...madness!

Date: Sun, 01 Oct 2006 18:28:18 +0200

Newsgroups: comp.lang.ada

> I think you *could* do away with GNAT project files, but that way lies madness.
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, programs, URLs, etc. and are updated regularly.

2007

January 03-06 Software Technology Track of the 40th Hawaii International Conference on System Sciences (HICSS-40), Waikoloa, Big Island, Hawaii, USA. Includes mini-tracks on: Software Engineering Decision Support (topics include: Design decisions; Reuse decisions; Maintenance decisions; Selection of software tool, methods or techniques; ...); Adaptive and Evolvable Software Systems (topics include: new strategies for improved modularization in order to support adaptations; ...); Components for Embedded and Real-time Systems (topics include: Component-based product lines for embedded applications; Real-time issues of component-based software engineering; Case studies and experience reports; ...); Visual Interactions in Software Artifacts; etc.

January 06-09 6th IEEE/IFIP Working Conference on Software Architecture (WICS'2007), Mumbai, India. Topics include: tutorial on Architecting Fault Tolerant Systems, etc.

January 14-16 8th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'2007), Nice, France. Co-located with POPL'2007. Topics include: program verification, program certification, abstract interpretation, static analysis, type systems, etc.


January 16 ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI'2007), Nice, France. Topics include: Typed intermediate languages and type-directed compilation; Type-based language support for safety and security; Types for interoperability; Type-based program analysis, transformation, and optimization; Dependent types and type-based proof assistants; Types for security protocols, concurrency, and distributed computing; Type based specifications of data structures and program invariants; Type-based memory management; Proof-carrying code and certifying compilation; etc.


January 17-19 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'2007), Nice, France. Topics include: fundamental principles and important innovations in the design, definition, analysis, transformation, implementation and verification of programming languages, programming systems, and programming abstractions.

January 20 POPL'2007 – Workshop on Programming Language Techniques for XML (PLAN-X'2007). Topics include: Design of programming and query languages for XML; Compilers and interpreters for XML-aware languages and optimization techniques; Languages and systems that can cope with XML fragments (messages) or very large XML instances (beyond main-memory size); Programming language glue between browsers, web services, and databases; etc.

January 20 2007 International Workshop on Foundations and Developments of Object-Oriented Languages (FOOL/WOOD'2007), Nice, France. Topics include: language semantics, type systems, program analysis and verification, concurrent and distributed languages, language-based security issues, etc.
January 22-26 **OOP'2007 Conference**, Munich, Germany. Topics include: requirements engineering, model-driven development, UML, Eclipse, security, etc., and applications in automation, automobile, medical, banking, telecommunications, etc.


February 07-09 **15th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP'2007)**, Naples, Italy. Topics include: Advanced Applications (scientific and engineering applications, multidisciplinary and multi-component applications, real-time applications, ...); Models and Tools for Programming Environments; Distributed Systems; Languages, Compilers and Runtime Support Systems (task and data parallel languages, object-oriented languages, dependability issues, ...); Parallel Computer Systems.

February 12-14 **International Conference on Tests And Proofs (TAP'2007)**, Zurich, Switzerland.


March 07-10 **38th ACM Technical Symposium on Computer Science Education (SIGCSE'2007)**, Covington, Kentucky, USA.


© Mar. 11-15 **Track on Object-Oriented Programming Languages and Systems (OOPS'2007)**. Topics include: Programming abstractions; Advanced type mechanisms and type safety; Multi-paradigm features; Language features in support of open systems; Program structuring, modularity, generative programming; Distributed Objects and Concurrency; Middleware; Heterogeneity and Interoperability; Applications of Distributed Object Computing; etc.

© Mar. 11-15 **Track on Software Engineering (SE'2007)**. Theme: "Developing Trustworthy Software Systems". Topics include: Trustworthy Software Systems Development; Software Testing, Validation and Verification; Model-Driven Architecture and Interface Design; Software Metrics, Cost Estimations and Benchmarking; Software Reuse and Component-Based Development; Real-Time Embedded Systems; Software Reliability Model and Implementation; Software Fault Tolerance and Software Availability; Reengineering for Safety and Security; etc.

© March 21-23 **2nd European Conference on Computer Systems (EuroSys'2007)**, Lisbon, Portugal. Topics include: All areas of operating systems and distributed systems; Systems aspects of: Programming language support, Parallel and concurrent computing, Dependable computing, Real-time and embedded computing, Middleware, Security, ...; etc.

March 21-23 **11th European Conference on Software Maintenance and Reengineering (CSMR'2007)**, Amsterdam, the Netherlands. Theme: "Software Evolution in Complex Software Intensive Systems". Topics include: software migration strategies and technologies, experience reports on maintenance and reengineering, etc.

March 24-April 01 **13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2007)**, Braga, Portugal. Part of ETAPS'2007. Topics include: rigorously based tools
and algorithms for the construction and analysis of systems; formal methods, software and hardware verification, static analysis, programming languages, software engineering, real-time systems, etc.

March 25-28

March 26-30
21st IEEE International Parallel and Distributed Processing Symposium (IPDPS’2007), Long Beach, California, USA. Topics include: Applications of parallel and distributed computing; Parallel and distributed software, including parallel programming languages and compilers, runtime systems, middleware, libraries, and programming environments and tools; etc.

March 26-27 15th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS’2007). Topics include: Applications and tools; Distributed real-time and embedded middleware; Soft real-time and mixed-critical systems; QoS based resource management and real-time scheduling; Programming languages and environments; Specification, modeling, and analysis of real-time systems; etc.

March 26-30 Workshop on Tools, Operating Systems and Programming Models for Developing Reliable Systems (TOPMoDeLS’2007). Topics include: Tools for recovery in parallel and distributed systems; Programming models and primitives for reliable distributed computing; Compilers for languages with primitives for reliability and recoverability; Compilers for domain specific languages with applications in distributed environments; Models for distributed systems; etc.

March 27-29 13th Conference on Languages and Models with Objects (LMO’2007), Toulouse, France.


April 17-19 13th International Real-Time Ada Workshop (IRTAW-2007), Woodstock, VT, USA. Topics include: early experiences in using Ada 2005 for the development of real-time systems and applications; implementation approaches for the new real-time features of Ada 2005; developing other real-time Ada profiles in addition to the Ravenscar profile; implications to Ada of growing use of multiprocessors in development of real-time systems; paradigms for using Ada 2005 for real-time distributed systems; definition of specific patterns and libraries for real-time systems development in Ada; how Ada relates to the certification of safety-critical and/or security-critical real-time systems; current ISO reports related to real-time Ada and new secondary standards or extensions; status of the Real-Time Specification for Java and other languages for real-time systems development, and user experience with current implementations and with issues of interoperability with Ada in embedded real-time systems; lessons learned from industrial experience with Ada and the Ravenscar Profile in actual real-time projects. Deadline for submissions: January 12, 2007 (position papers), March 16, 2007 (final paper).

April 25-27 Software & Systems Quality Conferences (SQC’2007), Duesseldorf, Germany.

May 07-09 10th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC’2007), Santorini Island, Greece. Topics include: Programming and system engineering (ORC paradigms, languages, RT Corba, UML, model-driven development of high integrity applications, specification, design, verification, validation, testing, maintenance, system of systems, etc.); System software (real-time kernels, middleware support for ORC, extensibility, allocation, scheduling, fault tolerance, security, etc.); Applications (embedded systems (automotive, avionics, consumer electronics, etc), real-time object-oriented simulations, etc.); System evaluation (timeliness, worst-case execution time, dependability, fault detection and recovery time, etc.); ...
May 20-26

29th **International Conference on Software Engineering** (ICSE’2007), Minneapolis, Minnesota, USA. Theme: "Developing Dependable Software".

May 22

1st **Workshop on Assessment of Contemporary Modularization Techniques** (ACoM.07). Topics include: Lessons learned from assessing new modularization techniques, Empirical studies, Comparative studies between new modularization techniques and conventional ones, Software metrics and quality models, etc. Deadline for submissions: February 1, 2007.

May 26

4th **International Workshop on Software Engineering for Automotive Systems** (SEAS’2007). Topics include: all aspects of software engineering for automotive systems, specifically all facets of integration of independently developed software parts to one system with emphasis on the following aspects: software quality, safety / reliability / robustness, component orientation in embedded systems, maintenance of the integrated embedded software system and compatibility of its components over the lifecycle, etc. Deadline for submissions: January 20, 2007.

May 27-30

7th **International Conference on Computational Science** (ICCS’2007), Beijing, China. Theme: "Advancing Science and Society through Computation".

May 28-31

5th **Object Oriented Technologies conference** (OOT’2007), Plzen (Pilsen), Czech Republic. Topics include: Software Engineering (software components, large-scale software, multi-language programming); Parallel and Distributed Computing (multithreading, distributed applications, ...); Programming Languages and Techniques (object-oriented techniques, programming paradigms, assertion support); Educational Aspects (teaching object-oriented paradigm, educational software); Software Security; Development on Different Platforms; Industrial Applications of Object Oriented Technologies; etc. Deadline for submissions: February 14, 2007 (abstracts), February 28, 2007 (papers).

May 29-Jun. 01

**Data Systems In Aerospace** (DASIA’2007), Naples, Italy.

June 06-08


June 09-16

3rd **History of Programming Languages Conference** (HOPL-III), San Diego, CA, USA. Co-located with FCRC’2007.

June 11-14

7th **International Conference on Algorithms and Architectures for Parallel Processing** (ICA3PP’2007), Hangzhou, China. Topics include: Distributed & Parallel Middleware, Parallel Programming Paradigms, Tools & Environments for Parallel & Distributed Software Development, etc.

June 14

PLDI2007 - ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS’2007), San Diego, California, USA. Topics include: the use of Programming Language and Program Analysis Techniques to improve the Security of Software Systems; Language-based techniques for security; Program analysis techniques for discovering security vulnerabilities; Specifying and enforcing security policies for information flow and access control; etc. Deadline for submissions: April 1, 2007.

June 18-21

**Systems and Software Technology Conference** (SSTC’2007), Tampa Bay, Florida, USA.

June 24-28

**Technology of Object-Oriented Languages and Systems** (TOOLS Europe’2007), Zurich, Switzerland. Topics include: all aspects of object technology and neighbouring fields, in particular model-based development, component-based development, and patterns (design, analysis and other applications); more generally, any contribution addressing topics in advanced software technology. Deadline for submissions: February 1, 2007 (technical papers), March 1, 2007 (workshops).
June 25-27 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'2007), Dundee, Scotland, UK.


June 25-29 27th International Conference on Distributed Computing Systems (ICDCS'2007), Toronto, Canada. Topics include: all aspects of distributed and parallel computing.

☺ July 01-02 12th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'2007), Berlin, Germany. Affiliated with CAV'2007. Topics include: Design, specification, code generation and testing with formal methods; Verification and validation of complex, distributed, real-time systems and embedded systems; Verification and validation methods that aim at circumventing shortcomings of existing methods with respect to their industrial applicability; Tools for the design and development of formal descriptions; Case studies and project reports on formal methods related projects with industrial participation (e.g. safety critical systems, mobile systems, object-based distributed systems); Application of formal methods in standardization and industrial forums. Deadline for submissions: March 30, 2007 (abstracts), April 6, 2007 (papers).

☺ July 05-08 6th International Symposium on Parallel and Distributed Computing (ISPDC'2007), Hagenberg, Austria. Topics include: Parallel Computing; Algorithms, Models and Formal Verification; Tools and Environments for Program Analysis; Task and Communication Scheduling and Load Balancing; Real-time Systems; Distributed Software Components; Real-time Distributed Systems; Security; Fault Tolerance; Applications and Case Studies; etc. Deadline for submissions: January 29, 2007.

☺ July 09-12 2007 International Conference on Software Engineering Theory and Practice (SETP-07), Orlando, FL, USA. Topics include: all areas of Software Engineering and all related areas, such as: Component-based software engineering; Critical and embedded software design; Distributed and parallel systems; Distribution and parallelism; Education (software engineering curriculum design); Embedded and real-time software; Empirical software engineering and metrics; Evolution and maintenance; High assurance software systems; Interoperability; Legal issues and standards; Object-oriented techniques; Program understanding issues; Programming languages; Quality management; Real-time software engineering; Reliability; Reverse engineering and software maintenance; Software architectures and design; Software components and reuse; Software cost estimation techniques; Software design and design patterns; Software engineering methodologies; Software engineering versus systems engineering; Software policy and ethics; Software reuse; Software safety and reliability; Software security; Software testing, evaluation and analysis technologies; Software tools and development environments; Survivable systems; Technology adoption; Verification, validation and quality assurance; etc. Deadline for submissions: February 1, 2007 (draft papers).


☺ Jul. 30-Aug. 03 21st European Conference on Object-Oriented Programming (ECOOP'2007), Berlin, Germany. Topics include: all areas relevant to object technology.

August 12-15 26th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'2007), Portland, Oregon, USA.

☺ August 28-31 13th International Conference on Parallel and Distributed Computing (Euro-Par'2007), Rennes, France. Topics include: the promotion and advancement of all aspects of parallel and distributed computing, such as support tools and environments, distributed systems, parallel and distributed programming, etc. Deadline for submissions: January 26, 2007 (full papers), April 2, 2007 (workshops).
September 03-07 16th International Conference on Parallel Architectures and Compilation Techniques (PaCT’2007), Pereslavl-Zalessky, Russia. Topics include: New trends and models in Parallel Programming; All aspects of the applications of parallel computer systems; Languages, environment and software tools supporting parallel processing; General architecture concepts, enabling technologies; Teaching parallel processing; etc. Deadline for submissions: January 20, 2007 (full papers), February 5, 2007 (extended abstracts).

September 04-07 International Conference on Parallel Computing 2007 (ParCo2007), Juelich & Aachen, Germany. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments. Deadline for submissions: March 4, 2007 (abstracts, mini-symposia), May 15, 2007 (presentations), July 31, 2007 (full papers).


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

2008

June 13th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE’2008), Madrid, Spain.
Using CORBA to Bring New Life to Legacy Ada: an Experience Report

Jean-Claude Mahieux, Bernard Maudry, Andrew Foster
PrismTech, Parc Fontaine de Jouence - 4 rue Angiboust 91460 - Marcoussis - FRANCE. email: jeanclaude.mahieux@prismtech.com

Abstract
In this short paper we report on a successful experience (of which we were happy protagonists) with the migration of a sizeable legacy Ada application to a new execution platform including the interaction with heterogeneous languages and components. We briefly illustrate the challenges we faced and the key choices we made to address them. We then conclude by drawing some lessons learned that could be of interest to other users.

1 Introduction
In the context of continuously evolving software technologies, the question of whether to invest in the preservation of existing Ada software often arises. This paper will describe how new innovative techniques that use CORBA can be deployed to extend the lifetime of an existing Ada system significantly and with low effort. The paper illustrates an example a combat management system developed completely in Ada. After describing the existing system and the key requirements placed on the upgrade of the system, we discuss how we devised the notion of a “rich IDL” to easily integrate CORBA in a legacy system.

2 Status and objectives
The existing system was configured as follows:

- Targeted to HpUX and HpRT
- Written in Ada83
- Using proprietary legacy middleware
- Subject to critical performance requirements.

The following key requirements were instead placed on the system upgrade:

- Retarget to Linux as the underlying operating system
- Migrate existing code to Ada95
- Maintain compatibility with the required external components via the legacy middleware
- Do not incur degradations of system performance

- Add an external Java GUI to facilitate access to large data structures
- Use COTS components whenever possible.

3 Technical issues
The application was first migrated to Ada 95. CORBA was immediately recognized as an obvious possible solution to share data between Java and Ada components in a standard manner. The PrismTech team (the authors) were invited to join the project as CORBA experts.

In the system in question the data structures that needed to be shared were huge, whereby system performance was a significant issue. It was therefore clear that converting types between the legacy world and the CORBA world should be avoided in so far as possible.

Our consideration of the fact that the system implementation was based on a dictionary of large Ada types, which was generated from an application specification lead us to the following conclusions:

- We should try to focus on typing, and try to replace current typing with an equivalent CORBA mapping. By doing this, we would be able to support the legacy Ada code without changes. An important implication in this regard was that any previous testing would still be valid.

- We should continue to exploit automatic code generation, not only for efficiency reasons, but also to ease integration in the application generation process.

4 Creation of a “rich” IDL
In order to meet these goals it was decided that the Ada types dictionary should be expressed in the CORBA IDL. To achieve this in a controlled fashion, we decided to implement an Ada to IDL translator.

Most Ada types could find an obvious IDL equivalent, including packages transposed to IDL modules.

For some Ada constructs that had no direct equivalence in IDL we introduced pragmas to preserve the Ada type information in the generated IDL files (e.g., subtypes, ranges, etc. ...)
Ada User Journal Volume 27, Number 4, December 2006

The output generated by the ada2idl translator is what we called a “rich” IDL, which provides all of the information we required to re-generate ready-to-compile Ada packages.

We illustrate this notion by relating a fragment of the original Ada code to the corresponding IDL definition:

```ada
package X is
  type T_Short is range -32768 .. 32767;
type Percent is new T_Short range 0 .. 100;
subtype Another_Short is T_Short range 0 .. 1000;
end X;
```

Below we show the output of the ada2idl translator:

```ada
module X {
typedef short T_SHORT ;
# pragma OrbAda_Directive
#   "Range" "T_SHORT  -32768    32767"
typedef T_SHORT   PERCENT;
# pragma OrbAda_Directive
#   "Range" " PERCENT  0  100"
# pragma OrbAda_Directive
#   "Insert_Line" "subtype
ANOTHER_SHORT is T_SHORT
range 0 .. 1000 ;"
}
```

5 IDL to Ada generation

The subsequent step was to enhance our idl2ada compiler to understand the pragmas retained in the input source, so that the generated Ada packages would embed all the original type details.

To illustrate what we wanted to achieve we show below a sample of the Ada code generated by our idl2ada compiler from the above IDL module.

```ada
package X is
  type T_SHORT is new Corba.Short
range -32768 .. 32767 ;
for T_SHORT'size use 16;
type PERCENT is new T_SHORT
range 0 .. 100 ;
for PERCENT'size use 16;
subtype ANOTHER_SHORT is T_SHORT
range 0 .. 1000 ;
end X;
```

6 Technical considerations and assessment

6.1 Automated control on new typing

To ensure that correct system behaviour was preserved, a test program was automatically generated, compiled and built with both the original and the newly generated types. Both versions of the test program were shown to generate an equivalent output, thereby proving that no property of the data types were corrupted by application of the ada2idl/idl2ada transformations.

6.2 Limitations in the rebuilt application

We did incur some limitations, though, which prevented us from achieving full preservation of properties. For example, some arrays where the indexes were enumerated types could not be expressed in IDL and therefore had to be expressed as standard arrays. In this case the only modifications that were necessary in the application code were with indexes ’pos and ’val. Fortunately, the number of lines to modify to address the problem was very small indeed and the application was thus easily rebuilt on top of the new types dictionary generated by the idl2ada compiler. As expected, the rebuilt application still ran correctly.

6.3 Addition of an interface

In order to make the application CORBA-compliant it was still necessary for us to equip it with a CORBA interface. This requirement was satisfied by providing a Java GUI with the ability to register a CORBA callback. By doing this the GUI would be notified automatically of each track update.

6.4 The middleware main loops

An important functional requirement on the resulting application was that is should manage CORBA messages in parallel with messages generated by the legacy middleware. To address this need, we contemplated two possible solutions:

- When the operation of the application is not thread-safe or else requires polling, then methods: ORB.work_pending and ORB.perform_work could be used in combination.
- Otherwise the ORB.run method will block the current thread and dispatch incoming CORBA requests.

The latter solution was used in the particular case of our project.

6.5 Use of IDL by Java/C++ environments

We are very well aware that any user-defined pragmas are not recognized by third-party idl compilers, and are consequently ignored. However, they are meant to be Ada specific and as such it is not required that they should be understood when compiling the idl to Java or C++ environments. For this reason and for those purposes idl compilers from other vendors can be used with the idl unmodified with no problem.
6.6 Robustness gained in the interface between Java/C++ and Ada applications

A side effect of supporting a “rich” IDL is that when CORBA requests from Java/C++ clients are dispatched to the Ada system with incorrectly initialized parameters, an Ada exception (“Constraint_Error”) is generated upon reading the request parameters. The exception is then caught by the CORBA library, which in turn returns a standard CORBA exception (“Impl_Limit”) to the client. This feature does indeed buy some increased robustness to the system in the regard of the interaction between heterogenous application components.

7 Summary of key migration points

Overall, we can summarize the key steps we took in performing the required migrations as follows:

- Develop new ad-hoc tools, such as an Ada to IDL compiler, to perform the automatic generation of type-safe IDL specifications for existing Ada interfaces
- Extend the CORBA IDL to more closely support the Ada type system. This provision will remove the need for inefficient data conversions at run time, thereby preserving system performance and also permitting legacy Ada interfaces to remain unchanged
- Add the CORBA bindings required to support the Ada type system.

Having taken those steps, permitted us to exploit a vast span of automatic code generation techniques, which made the task of migrating a huge amount of legacy code dramatically more efficient.

Furthermore, being able to support the legacy Ada code without changes also meant that any previous testing was still valid.

8 Conclusion

By following the approaches and strategy outlined above, the existing Ada combat management system was successfully upgraded with minimum cost and, most notably, within project deadlines.

Consequently, as a result of the project effort, the legacy Ada application had been “CORBA enabled”, while at the same time, the original Ada interface had been fully retained.
The Publisher Framework

Judith Klein¹, Drasko Sotirovski²

¹ Lockheed Martin, 9211 Corporate Boulevard, Rockville, MD 20850, USA. email: judith.klein@lmco.com
² Raytheon Canada Ltd, #150-13375 Commerce Parkway, Richmond, BC, V6V 2L1, CANADA. email: drasko@acm.org

Abstract

One of the lasting challenges in building distributed fault tolerant systems is keeping application code size and complexity down. This can be done by capturing the nuances of distributed computing environment and redundant fault tolerant elements into a common infrastructure layer, thus factoring the code that would otherwise need to be written again and again by each distributed fault tolerant software component. When the application code has many complexities, and Air Traffic Control (ATC) is certainly one such example, achieving this goal becomes paramount.

Under a project called En Route Automation Modernization (ERAM), the Federal Aviation Administration (FAA) is developing a replacement for its aging en route assets. At the same time, a foundation is being created for the anticipated future enhancements, driven by the projected increase in air traffic. At the core of the ERAM design is a distributed object oriented (OO) framework called Publisher FrameWork (PFW), which is ERAM’s answer to the aforementioned OO challenge. This paper describes the PFW properties, the experiences with it accumulated through the first build of the ERAM program, and its applicability to fault tolerant computing.

1 Introduction

Distributed computing is 20+ years old, but it only took the main stage with the explosion of networking and the internet in particular. The outburst of distributed computing frameworks (CORBA, J2EE, .NET, to name just the few most popular) is no surprise and one should expect a plethora of distributed computing environments before some of the difficult issues in distributing computing are settled satisfactorily. From this point of view, PFW is no exception: just one of the many! Although true to an extent, PFW is also more: an attempt to raise the bar and attack not only the relatively simple issues of messaging/dispatching, but also some rather difficult issues related to encapsulation, extensibility, scalability, performance, and availability, all with no significant increase in application code size and complexity. PFW is a lightweight framework: it enables applications to focus solely on the application domain.

In the subsequent sections, we describe the software component methodology we have followed and the resulting need for publication services, mirror storage and subscriber synchronization. To keep our experience report focused, we are not going to compare PFW with CORBA or J2EE, which have since started including fault tolerant elements. We hope to provide enough insight into PFW for the readers to make that comparison on their own. The fact that PFW provides cross-language support (ERAM components are split between Ada and C++) was also ruled outside the scope of this particular discussion.

2 Distributed Software Components

Our expertise is in building large scale, distributed, fault tolerant, near real-time systems. The application domain we’ve been concentrating on for over the last 15+ years has been air traffic control. We, at Lockheed Martin, had already developed and fielded a robust infrastructure called FlightDeck™ (see Figure 1), which provides a rich set of relevant services. We proceeded to concentrate on the domain of air traffic control. Here we started with a data model and ended with a set of software components, a description of how the components should behave, with a common behavioral pattern and a strict dependency hierarchy for ease in building the system. This inevitably led to PFW, an infrastructure extension which captures and reinforces this pattern.

3 Component Definition

A component is a logical grouping of software whose definition is based on the problem domain. A software component provides a cohesive set of services, exports a well defined interface (application programming interface, API); it encapsulates implementation details (internal databases, data structures and internal functions are hidden from the client). Furthermore, software components can be independently developed and tested.

3.1 Methodology Used to Define Components

Objects in the problem domain (e.g., airport, route, target report) form natural dependencies and relationships (e.g., a flight plan has a departure point, a destination and a route). Following this well established software engineering principle, we grouped cohesive objects into software components in a manner in which interfaces between components were minimized. In the process, we established dependency rules among components: a hierarchy of components which is strictly enforced in the build process.

¹ FlightDeck™ is a trademark of Lockheed Martin.
² One of us, while working on the Canadian Automated Air Traffic System (CAATS), developed a common middleware layer with similar properties to PFW (see references [1] and [2]), leading us to believe that these system types are conducive to such software designs.
Communication services, featuring:
- broadcast, multicast, and connected data transfer with support for redundant servers
- automatic division of messages to network frame size
- naming services which isolate applications from the system topology
- network overload protection
- higher level abstractions are also supported
  - publish/subscribe
  - clear/servers
- reduced overhead with minimal data movement

Availability services, featuring:
- detection of software crashes (<1 ms) and hangs (adaptable by application)
- detection of processor failures (<1 sec, adaptable)
- recovery models (adaptable) which can be initiated either locally, within server groups, or across the system (<1 sec, adaptable)
- hardware certification at IFL and at adapted intervals

System time services, featuring:
- time of day clocks
- multiple simulated clocks
- highly synchronized system clock (20 ms or better, adaptable)

Figure 1: Side-Bar on FlightDeck Middleware.

System recording/analysis, featuring:
- online and offline analysis tools
- adaptable, commandable, and event driven data collection

System management services, featuring:
- centralized monitor and control capability with multiple command nodes and verification methods, and hierarchical status
- low impact distribution using point broadcast to allow concurrent mission functions
- checkpointing during distribution and processor IFL
- management of multiple releases

Application builders, featuring:
- event services, which provides an integrated interface to the system communication and time services
- data checkpointing services
- synchronized distribution of shared memory to selected nodes
- structured support for command and control of state and location of user-defined system elements, each of which may be independent or part of a complex hierarchy of elements related via a set of Boolean operators
- ready application support for redundant applications

Figure 2: FAA En Route Air Traffic Control.

The current Host Computer System operating in the Federal Aviation Administration (FAA)'s En Route centers has been the backbone of the US National Airspace System (NAS) for thirty-five years. In 2003 the FAA awarded a contract to Lockheed Martin to replace the existing NAS Host and DAF (Direct Access Radar Channel)

ERAM adds improved capabilities of NAS Architecture, Free Flight Initiatives, Advanced Communication, Navigation, Surveillance, Information Management, and Decision Support Technologies that can now be applied to Air Traffic Management. It will replace the existing mainframe-centric Host architecture with a state of the art open and supportable environment.

There currently are twenty en route centers in the Continental U.S. A center can have hundreds of air traffic controllers working side by side in some combination of Radar, Data, Assistant and Specialist positions. Numerous external interfaces, including other en route centers, send a constant stream of inputs and expect similar data. A number of data servers perform the bulk of the algorithmic processing on behalf of one en route center; they are all redundant for fault tolerance reasons. Workstations are dedicated to the controllers. Throughput and response time requirements place challenging constraints on the system architecture and design.
Component definitions are advertised in terms of provided services on the encapsulated data objects. Added benefits of this approach are: extensibility (e.g., additional components are built using existing components, promoting re-use of existing components) and ease of component replacement (as long as the API is invariant, the implementation can change).

4 Component Deployment to Physical Nodes

An executable is a physical grouping of software, an independently start-able, stoppable program: with Unix, this is a single process that may includes multiple threads. Definition of executables is based on the knowledge of the physical architecture and with fault tolerance in mind: the executable is a unit of failure, while a thread is a unit of concurrency. Executables we build in the air traffic control domain, follow the event-driven model:

a) Multiple threads of execution are packaged into the same executable.

b) Each thread of execution is in a forever loop waiting for events, servicing each event in priority order, making synchronous (e.g., library calls) and asynchronous service requests (e.g., to a service residing on a different node).

c) For asynchronous requests the address of a callback procedure is provided to the service: when the service completes, the callback is invoked with the results.

A component spans executables. Parts of a component can be bound into a server/publisher executable resident on one node, while other parts of the same component are bound into a client/subscriber executable resident on a different node (this is similar to the J2EE concepts of local and remote interfaces or CORBA proxies and skeletons). The data exchange between the publisher of the component and the subscriber is internal to the component: the format of that exchange can be modified without impacting the users of the component’s services (since they access the component strictly by using the API); in other words, whether the data is encoded for transmission (a.k.a. serialized) into XML format, binary format, or something else, the users of the components are unaffected as long as the APIs used to access the objects (attributes, methods) are constant. Publishers of multiple components can be bound together into a single executable, along with a number of other components’ client-side code to achieve the application mission (e.g., detecting conflicts between aircraft).

We describe the physical software architecture by showing the placement of defined executables on nodes (processors) of the hardware architecture and by showing the parts of components that were bound together to form each executable.

Lockheed Martin defined and implemented a first version of the component model (i.e. the collection of components needed to implement an air traffic control system, along with their interfaces and interactions) for use in “User Request Evaluation Tool”. This tool is now deployed nation-wide; it automatically predicts and notifies controllers of conflicts between aircraft or special activity airspace. The system also allows controllers to quickly determine whether proposed flight path changes will conflict with en route traffic or airspace. By allowing controllers to evaluate route change requests and to assign conflict free routing, the airspace users are able to save both time and fuel.

For the first implementation each component’s behavior was described from an architecture standpoint – the pattern was defined, but no common framework was provided. The resulting implementation proved that the concepts were solid. However, we noticed that there was large variation in the implementation specifics of components, as well as some degree of code duplication for common component behavior. We concluded that benefits could be reaped from factoring out the common behavior into a common framework:

a) Overall code size could be reduced.

b) Errors in implementing the basics of the component framework could be eliminated when correcting them once in the common framework.

c) Components would be more maintainable since they would be concentrating on the domain expertise rather than on framework matters.

Such observations led us to the development of the Publisher FrameWork (PFW) on which most components in the air traffic control domain of ERAM are now built. As we prototyped PFW, we found more and more common behavior to be factored and included into PFW, such as data redundancy for fault tolerance.

5 The Publisher Framework

The Publisher FrameWork (PFW) provides a framework for uniform, consistent development of software components. The design pattern (see Figure 3) implements support for:

a) A server to publish objects to subscribers and to process requests from clients.

b) An agent acting as a local subscriber to receive published objects, translate them into messages and multicast them to all remote subscribers. The use of multicast mechanism makes PFW scalable to the hundreds of positions that must be supported in an en-route center.

c) A proxy to receive multicast messages, translate them back into objects and republish them to local clients. The component user, when notified that an update has arrived, is guaranteed that the mirror is

---

3 Some legacy components were not converted to using PFW to minimize change of working components.
current, i.e., the update has been applied to the content of the mirror; therefore the component user can make queries against the object attributes from within the context of the call-back. Additionally, the proxy facilitates requests from the clients to the server; a watchdog timer is used to monitor the arrival of a timely reply from the Server – the client is therefore guaranteed to be notified either about the completion of the request or about its timeout.

d) A mirror to augment the proxy by retaining a copy of the data published by the server for use in local queries. The existence of the mirror provides the client with the convenience of accessing the data not only when the information is received, but also as part of other processing, such as the expiration of a timer. In response to a request to update a server object, the mirror is updated before the confirmation is delivered to requestor, so that the requestor can reach into the mirror and access object attributes and methods with the assurance that the object is up-to-date.

Figure 3: Anatomy of PFW distributed service.

In ERAM, for which PFW was developed, all the software components are known – their names (server name, published data-stream name) and APIs are documented; all necessary discovery is done by infrastructure alone, which locates the registered server(s), selects the Primary executable of that server, and delivers client requests to the server.

Note that there is a plethora of requirements implemented by PFW on behalf of all components that are less interesting to describe in this paper, yet helpful to the component implementers just the same; one example is invoking the recording service and error reporting service (to log commonly recorded events and data) for detected errors.

5.1 Mirror and Original, Queries and Updates

As introduced above, a mirror encapsulates client-side storage of object replica matching the original objects of the server’s internal database. In other words, the client-side proxy subscribes to (registers interest in) the data-stream published by the server; when a publication is received, the proxy first updates the object replica in the mirror storage; other local clients can then safely be invoked with the guarantee that the mirror is current and consistent with the server’s internal database. Having a mirror presents the advantage of being able to perform synchronous queries against the local object cache.

Requests for update of an object are asynchronous: the request is forwarded to the server (whether local to the executable or remote); a call-back procedure is provided so that it can be invoked when the results of the asynchronous update are received. All updates to an object are performed only by the owner of the object (the server/publisher, not a mirror – merely a copy of the object is stored in the mirror).

---

4 See details on Primary vs. Standby in Figure 1: Side-Bar on FlightDeck Middleware.
A simple and robust approach insures consistency of the data throughout.

Finally, local clients can be notified of changes (in addition to being able to query the local mirror storage). There are two kinds of registration:

(a) For all objects of a class, resulting in the registrant being notified whenever an object is created (added to the mirror storage), deleted (removed from the mirror storage) or updated (modified in the mirror storage);

(b) For a specific instance, resulting in the registrant being notified whenever that instance is modified, including deletion.

In conclusion, the object replica PFW maintains in the mirror storage is, from a client’s perspective, virtually indistinguishable from the original: it can be observed through registration/notification, queried and updated – albeit in an asynchronous fashion. This provides near-perfect object location transparency. Only the fact that update methods are asynchronous hint to a client that the target object may or may not be local. In any other respect, the client-side replica is indistinguishable from the original object.

5.2 Fault Tolerance
A primary-standby pattern is a standard arrangement for high integrity systems.

The goal is always one and the same: to provide full protection from hardware faults and protection from at least transient software faults. In industrial applications, like ATC systems, the failure model is always fail-safe, but its dynamic characteristics vary from component to component, depending on the component criticality and the required switchover time. Components in ERAM range from active fail-silent (all redundant elements are active and at all times ready to provide services) to passive fail-stop (redundant elements are available but become active only after the primary had failed). In other words, primary provides the service with a (more-or-less hot) standby ready to replace it in case of a failure. In addition, if a service fails when no standby is available, the service can be restored with little or no loss in functionality from the checkpoint data that the primary and standby have saved on disk.

When prototyping PFW, we concluded that we can include support for the required spectrum of Standby designs by simply making the Standby a Subscriber to the very data-stream its Primary is publishing, as described in Figure 4. The essence of PFW fault tolerant behaviour, from a client perspective, is best described by simply stating: the client side object replicas are transparently rewired from the originals in the failed primary to the new originals in the new primary. This is however, as many a reader will know from experience, easier said than done. PFW implements near perfect transparency relative to both outstanding requests and clients registered for notification. In the following paragraphs, we describe some of the usual yet intricate obstacles in achieving these goals (and give the meaning of ‘near-perfect transparency’, ‘more-or less hot standby’, etc.).

Not only redundant servers fail, but clients can fail too. Having no redundancy, such computing elements need to get back in sync with the rest of the distributed environment, which poses additional challenges; in particular for getting the new members back in sync with no adverse performance impact on the system. PFW approach to solving this issue is deemed outside the scope of this particular discussion and readers interested in it are referred to [3].

5.3 Enhancements under Consideration
At the time the primary service failed, it may have been processing a request from a client, with more requests stuck in its input queue. To make the switchover transparent to clients, all these requests need to be processed, with the new primary taking over where the old primary stopped.

Obviously, the request being processed at the time the Primary failed may be the killer request and automatically reprocessing it exposes the system to the common mode failure. All other requests are safe to reprocess (in particular if addressed to other object instances). PFW currently makes clients with outstanding requests decide. Clients will receive a time-out on the request and must decide to resubmit it or not. PFW improvements under consideration include a means for clients to find out if the request is suspected to be the killer (i.e. if this very request was processed when the primary failed) as well as automatic resubmission of the requests stuck in the input queue of the primary at the time it died, provided they are not directed to the same object as the killer request. See points 1, 2 in Figure 4 PFW Facilitated Switchover of a Component.

Objects in the server are not only state; in addition to the state, an object may have some dynamic context, e.g. an outstanding timer to do something. This dynamic context can of course be mirrored by the hot standby. However, this makes primary and standby go through the same computational history and increases the likelihood of simultaneous primary-standby failure. For this reason, many system designs (including PFW) opt for recreating this dynamic context upon switchover. So far so good, but processing requests is only possible after the dynamic context is re-established. Yet re-establishing the dynamic context for thousands of objects can take a significant amount of time resulting in an unacceptable hiccup in system performance immediately after a switchover. PFW’s answer to this significant challenge is described in Figure 5 at point points 3, 4.
6 Conclusions

High availability systems have needs above and beyond the functionality provided by the contemporary commercial middleware (CORBA, J2EE, .NET, to name a few). ERAM successfully serves these needs through PFW by extending the middleware services to provide location transparency and fault tolerance. As additional common behavioral patterns become apparent, the authors will consider incorporating capabilities to address these into future PFW releases. In return, an important part of common and intricate implementation is factored while leaving the application domain to remain focused on solving domain issues.

Acknowledgements

The authors acknowledge the following people who made significant contributions to the concept definition, prototype, design and development of PFW:

- Tim Donovan, Software Architect with Raytheon Integrated Defense Systems, Tewksbury, MA
- Sam Carnicelli, Chief Designer with Lockheed Martin Transportation and Security Solutions, Cato, NY
- This work was performed under the contract from the Federal Aviation Administration, DTFA01-03-C-00015. The support and review from Jeff O’Leary, FAA ERAM Product Team Software Lead is especially noted and appreciated.

References

Figure 5: PFW facilitated switchover of a component.

(1) At the time primary server fails, it may be processing a request from one of the clients with other requests waiting in its inbound queue.
(2) Secondary server, after FlightDeck had rewired the communication, publishes the killer request so that all clients with outstanding requests can resubmit them – except for the client holding the killer request.
(3) Secondary is using the system idle time to recreate the dynamic context of all objects, a quantum at a time.
(4) A request received for an object that has not been restored yet triggers the recreation and only then proceeds to executing the request.

About the Authors

Judith Klein is a certified systems architect at Lockheed Martin Transportation and Security Solutions. She has 28 years of experience developing distributed real-time systems of various sizes in different domains; the last 15 years have been focused on air traffic control. She has a BS in applied mathematics and computer science from Carnegie Mellon University in Pittsburgh, PA and an MS in technical management from Johns Hopkins University in Baltimore, MD. She is a senior member of the IEEE and a member of the Association for Computing Machinery.

Drasko Sotirovski is a software architect at Raytheon Systems Canada. He has 27 years of experience in developing large-scale real-time software for defence, simulation, transport, and telecommunication systems for several European and North American customers. His research interests are software architecture and distributed object-oriented technologies. He received a BSc in technical physics and computer science from Elektrotehnicki Fakultet u Beogradu, Yugoslavia. He is a member of the IEEE Computer Society and the Association for Computing Machinery. He is also a PEng with the new CSED (Computer and Software Engineering Division) branch of APEG BC.
Ada-Europe 2006 Sponsors

AdaCore
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@adacore.com URL: www.adacore.com

Aonix
Contact: Jacques Brygier
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.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: info@greenhills.com URL: www.greenhills.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

Praxis High Integrity Systems Ltd
Contact: Rod Chapman
20 Manvers Street, Bath, BA1 1PX, UK
Tel: +44-1225-466-991 Fax: +44-1225-469-006
Email: sparkinfo@praxis-his.com URL: www.sparkada.com

Ellidiss Software
TNI Europe Limited
Contact: Pam Flood
Triad House, Mountbatten Court, Worrall Street, Congleton, CW12 1DT, UK
Tel: +44-1260-29-14-49 Fax: +44-1260-29-14-49
Email: info@tni-europe.com URL: www.ellidiss.com