ADA USER JOURNAL

Volume 27
Number 3
September 2006

Contents

Editorial Policy for *Ada User Journal*

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

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

Reprinted Articles

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

Editorial

First of all I want to apologize to our readers for the late arrival of this issue of the journal to their doorstep. The production of AUJ 27-3 encountered very serious and equally unwelcome technical problems that took us time and effort to overcome, which our tight agendas have very little provisions for. My gratitude goes to Santiago Urueña, our News editor, who was the one hit the most by my friend Murphy and yet survived the hit, and to Dirk Craeynest, our long-time Calendar and Forthcoming Events editor for reliably providing a steady flow of material for the journal.

Lateness of the September production aside, the end of Summer brough us the very good news that the Ada 2005 amendment had gained the technical approval of ISO/IEC JTC 1/SC 22, thus freezing the new technical shape of the Ada language. That was a major achievement for the Ada community, which we celebrate in this issue by hosting an interesting retrospective reflection by Pascal Leroy, the chair of the group of technical experts tasked to shape the new version of the language. Well done, Pascal.

With this issue we also begin the publication of summary articles from the industrial presentations delivered at the Ada-Europe 2006 conference that took place in Porto back in June. The first author to contribute was Dave Kleidermacher, who shared with us some guidance for the rapid production of reliable software. Thanks Dave.

We are also happy to share with our readers the announcement that the 13th edition of the International Real-Time Ada Workshop (better known as IRTAW) will take place in April 2007. A welcome return for an event that has always made important contributions to the progress and fostering of the Ada language in the domain of real-time and high-integrity systems. You will find the details on page 176. The return of IRTAW means that the year 2007 will enjoy at least three major Ada-related events, which sounds like good news to us all.

Regrettably this editorial has to close on a negative tone, for which I apologise to the readership. Thanks to the advice of one of our readers we have in fact sadly realized that the article entitled "*A New Strategy Pattern for OO Technology*" and published on pp. 110 – 116 of AUJ 27-2, appears to be a case of serious plagiarism in that it lifted verbatim text and illustrations from the book: Head First Design Patterns, authored by Eric and Elizabeth Freeman and published in October 2004 by O'Really associates with ISBN 0-596-00712-4. Our humblest and most sincere apologies to the original authors. As for the perpetrators of the fraud, who were not responding to our complaints in the regard of this most annoying incident, we say that they taught us the hard way a good lesson into tightening up our review process. We shall strive to keep this promise in the future.

Tullio Vardanega
Padova
September 2006
Email: tullio.vardanega@math.unipd.it

News

Santiago Urueña

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

Contents

Ada-related Events

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

Jun 20 — Ada-Belgium General Assembly

From: Dirk Craeynest <dirk@apollo.cs.kuleuven.ac.be>
Organization: Ada-Belgium, c/o Dept. of Computer Science, K.U.Leuven
Date: 1 Jun 2006 01:33:49 +0200
Subject: UML2 profile enforcing Ravenscar model, Tue 20 Jun 19:45, Ada-Belgium
Newsgroups: comp.lang.ada,fr.comp.lang.ada,be.comp.programming,nl.comp.programmeren

Ada-Belgium Special Evening Event

Ada-Belgium is pleased to announce our next event: a technical presentation by Tullio Vardanega of the University of Padua, Italy.

Correctness by constr uction: a UML2 profile enforcing the Ravenscar Computational Model

Tuesday, June 20, 2006, 19:45-21:45

at the U.L.B., Department of Computer Science, Campus de la Plaine, "Forum" complex, auditorium "Forum D", Boulevard du Triomphe / Triomflaan, B-1050 Brussels (after the Ada-Belgium 2006 General Assembly at 18:45). http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/local.html

Announcement

Ada-Belgium will hold its 13th annual General Assembly on Tuesday, June 20, 2006, at the U.L.B., Department of Computer Science, Boulevard du Triomphe / Triomflaan, B-1050 Brussels, at 18:45. The official convocation is available, also in PDF format, and is sent via postal mail to all members.

There will be refreshments and pizza for Ada-Belgium members at 18:00. Please notify us if you are a current or new member and intend to participate at this informal "pre-meeting".

At 19:45 the General Assembly will be followed by a technical presentation on "Correctness by construction: UML2 profile enforcing the Ravenscar Computational Model", by Tullio Vardanega from the University of Padua, Italy.

Abstract

In this talk we illustrate the results of a research project that attempts to unite three distinct fronts of advances in the engineering of high-integrity software systems:

* the pursuit of correctness by construction;

* the reliance on the UML2 notions of profile supported by meta-model ontologies;

* the adoption of an educated approach to the use of concurrency by compliance with the computational model entailed by the Ravenscar Profile.

The talk will proceed in three successive steps:

* we first discuss how an initial UML2 profile can be built by mapping the basic ontologies of HRT-HOOD onto the UML2 meta-model;
* subsequently we address and overcome some frustrating limitations inflicted by the HOOD heritage and make an important step towards better integration of the HRT and OO dimensions of modern systems;

* finally we show how the increased expressive power of Ada 2005 permits to greatly amplify the benefits of correct-by-construction model-based code generation via factorization and instantiation.

Speaker

Tullio Vardanega, from the Department of Pure and Applied Mathematics of the University of Padua, is an expert in the development of real-time embedded systems, and processes and methodologies for the engineering of software-intensive high-integrity systems. Before joining the University of Padua, he worked for a long time at ESA, the European Space Agency.

He is Ada-Europe Board member and Editor of the Ada User Journal. He is active in ISO's Ada standardization working group (WG9), more specifically in the Ada Rapporteur Group (ARG, language maintenance) and the Annex H Rapporteur Group (HRG, guidance for high integrity applications in Ada).

Participation

Everyone interested is welcome at either or both parts of this meeting. As usual, the event is free and presentations are in English.

If you plan to attend the General Assembly or the technical presentation, we would appreciate it if you could inform us by e-mail (please also specify if you intend to participate at the informal "pre-meeting"). Although no formal registration is required, this helps our preparations.

All Ada-Belgium members have a vote at the General Assembly, can add items to the agenda, and can be a candidate for a position on the Board (see the convocation for more details).

If you are a member but have not yet renewed your affiliation please do so by paying the appropriate fee before the General Assembly (you have also received a printed request via normal mail). If you are interested to become a new member, please register by filling out the 2006 membership application form and by paying the appropriate fee before the General Assembly. After payment you will receive a receipt from our treasurer and you are considered a member of the organization for the year 2006 with all member benefits. Please settle this invoice ASAP. Early renewal ensures you receive the full Ada-Belgium membership benefits (including the Ada-Europe indirect membership benefits package).

Directions

This event takes place at the premises of the Universit Libre de Bruxelles (U.L.B.), Boulevard du Triomphe / Triomflaan, Campus de la Plaine, 1050 Brussels, Belgium. Exact location is auditorium "Forum D", in the "Forum" complex.

An access plan to the Campus de la Plaine of the U.L.B. is available. Parking facilities are at access no. 2 (parking Fraiteur) or no. 4 (parking UAE, the closest, usually has free space). You can check on-line how to reach the Campus de la Plaine by car or by public transport.

Looking forward to meeting many of you in Brussels!

Dirk Craeynest

President Ada-Belgium

Dirk.Craeynest@cs.kuleuven.be

Acknowledgments

We would like to thank our sponsors for their continued support of our activities: AdaCore, Katholieke Universiteit Leuven (K.U.Leuven), Offis nv/sa — Aubay Group, and Universit Libre de Bruxelles (U.L.B.).

[See also same topic in AUJ 26-2 (Jun 2005), pp.69–70. —su]

15 Nov — SIGAda Award Nominations

From: John McCormick <mccormick@cs.uni.edu>
Newsgroups: comp.lang.ada
Subject: Call for SIGAda Award Nominations
Date: 11 Sep 2006 10:57:53 -0700

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 same topic in AUJ 26-4 (Dec 2005), p.229. —su]

Ada and Education

Public Ada 95 Courses

From: Ed Colbert <colbert@abssw.com>
Date: 8 Jun 2006 13:05:52 -0700
Subject: [Announcing] Public Ada 95 Courses 17-21 July in Carlsbad CA
Newsgroups: comp.lang.ada

Absolute Software will be holding a public Ada 95 course during the week of 17 July 2006 in Carlsbad, CA. You can find a full description and registration form on our web-site, www.abssw.com. Click the Public Courses button in the left margin. (We also offer courses on software architecture-based development, safety-critical development, object-oriented methods, and other object-oriented languages.)

[See also same topic in AUJ 26-3 (Sep 2005), pp.150–151. —su]

Ada 2005 Upgrade Course

From: Jean-Pierre Rosen <rosen@adalog.fr>
Newsgroups: comp.lang.ada
Subject: Re: Ada 2005 courses?
Date: Mon, 26 Jun 2006 09:47:20 +0200
Organization: Adalog

> With Ada 2005 ISO approval almost assured are their any classes planned that cover the new features?

Adalog is offering an Ada 95->Ada 2005 upgrade course (see http://www.adalog.fr). The scheduled one is in French, but we can make it in-house in English.

Please get in touch with me if you need further information

Ada-related Tools

PragmARC — PragmAda Reusable Components

From: PragmAda Software Engineering <pragmada@mchsi.com>
Organization: PragmAda Software Engineering
Date: Thu, 27 Apr 2006 19:42:03 GMT
Subject: Announcement: New Version of the PragmAda Reusable Components
Newsgroups: comp.lang.ada

PragmAda Software Engineering announces a new release of the PragmAda Reusable Components. This release adds PragmARC.Genetic_Algorithm, a generic framework for genetic programming.

The PragmARCs are available from the PragmAda web site:

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

Error reports, comments, and suggestions are always welcome.

[See also same topic in AUJ 27-1 (Mar 2006), pp.7–8. —su]

Simple components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple components v2.3
Newsgroups: comp.lang.ada
Date: Tue, 15 Aug 2006 21:47:06 +0200

The new version is here:

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

Changes to v2.2:

- Generic package Object.Handle.Generic_Handle_Set is provided for sets of objects accessed through handles;

- Insert procedure was added to Generic_Set with a parameter to determine whether the item has been added to the set:

- The package Parsers.Multiline_Sources.Standard_Input was added to provide interface to the standard input file;

- Packages for dealing with XPM image format were added;

- Bug fix in Stack_Storage;

- Bug fix in the implementation of "and" in Generic_Set and Object.Handle.Generic_Set.

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

Auto_Text_IO & SAL

From: Stephen Leake <stephen_leake@acm.org>
Date: Sun, 18 Jun 2006 17:17:05 -0400
Subject: SAL, Auto_Text_IO release
Newsgroups: comp.lang.ada

I've downloaded GNAT GPL-2006, and it compiles all of my SAL and Auto_Text_IO code, and passes all tests, without problems. I think this is a first for a public GNAT release!

So I'm releasing a new version of SAL (2.00) and Auto_Text_IO (3.03). Support for GNAT 3.15p is now removed, making some things simpler; I've started using some Ada 2005 features (mainly 'raise … with <string>;'.

There are lots of improvements in SAL; it's been almost two years since the last release, and I've been improving it a lot for work. The biggest additions are support for left- and right-multiply quaternions (a somewhat obscure topic, but it took a lot of effort :), and many more features for config files.

Auto_Text_IO hasn't changed much.

See http://www.toadmail.com/~ada_wizard/ for more info.

[See also same topic in AUJ 25-4 (Dec 2004), p.193. —su]

GNU Ada Compiler

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: [gnuada] R5 release with lots of improvements now available
Date: Sun, 09 Jul 2006 11:13:17 +0200
Newsgroups: comp.lang.ada

With the R5 release many loose ends have been tied up:

1) The package naming convention changed to closer reflect the naming convention by the distribution.

2) The package dependencies have been properly filled in so useful error messages for missing packages are given.

4) The source packages are now self contained so that "rpmbuild --rebuild xxxx.src.rpm" should now work [1].

3) Only one set of source packages are provided.

5) RPM-Build-Script are now fully compatible with GNAT/Pro.

6) Configurations Scripts now updated.

And on top of all these improvements both GCC and GPL are created with the newest releases from the FSF and AdaCore.

All in all, this release is a lot better than any previous release.

[1] you still need an appropriate ~/.rpmmacros file.

[http://gnuada.sourceforge.net —su]

[See also same topic in AUJ 27-2 (Jun 2006), p.71. —su]

GNU Ada MinGW

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: [gnuada] MinGW cross compiler available
Date: Tue, 22 Aug 2006 19:06:34 +0200
Newsgroups: comp.lang.ada

MinGW has been our problem child for quite a while but now we have a MinGW cross compiler ready for your test.

Currently available as Build-Hosts are SuSE Linux 10.1 X86_64 [1] and MS-Windows Cygwin [2]. Look out for the "gnat-mingw-*" files.

The Toolchain only contains Ada and C as the other languages are not cross compile and/or windows friendly.

And yes, we do hope to use one of the cross compilers to create a native MinGW compiler. Details on how to use the compiler will be added to the MinGW wiki page [3] when I find some time.

[1] http://gnuada.sourceforge.net/pmwiki.php/Install/SuSE

[2] http://gnuada.sourceforge.net/pmwiki.php/Install/Cygwin

[3] http://gnuada.sourceforge.net/pmwiki.php/Install/MinGW

GNU Ada GPS

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: [gnuada] Finaly: GPS 4.0.0 available.
Date: Sat, 09 Sep 2006 20:19:11 +0200
Newsgroups: comp.lang.ada

Finally after many unsuccessful attempts we have created a working GPS. And not any old one — the very current GPS 4.0.0 with lots of new features.

"GNAT/GPL SuSE 10.1 x86_64" and "GNAT/GPL Source" are uploaded and others will follow when they become available.

Please note that you can use the GPL version of GPS together with the GCC version of GNAT — just the setup is a bit more tricky but you won't need to install two GPS if you don't want to.

[http://gnuada.sourceforge.net/ —su]

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: Re: Finaly: GPS 4.0.0 available.
Date: 12 Sep 2006 03:56:14 -0700
Newsgroups: comp.lang.ada

> I'd like to try this on SuSE 10.1_586 but since I've been away from Ada for a while could someone please tell me where this downloadable from.

For SuSE 10.1 568 it should be fairly simple as SuSE 10.1 x86_64 is already there. You need the base setup:

http://gnuada.sourceforge.net/pmwiki.php/Packages/GNATConfig

http://gnuada.sourceforge.net/pmwiki.php/Packages/GNATRPM

Then download the Source RPMs:

http://sourceforge.net/project/showfiles.php?group_id=3D12974&package_id=3D=191854

After that the fun of building begins:

http://gnuada.sourceforge.net/pmwiki.php/RPM/HomePage

Now there is one hurdle here: You need the GPL version of GNAT for the GPS (unless you want to patch the sources). But the GPL version can only be build with another GPL version.

For Linux 586 this is fairly simple: download a GPL version from Libre. For SuSE Linux you you can also download and install the SuSe 9.2 version to create a 10.1 version.

Initial setup is hard but once everything is set up then creation is as simple as calling:

```
Package_Step1.bash
Package_Step2.bash
Package_Step3.bash
```

or

```
make gpl
make gpl-gktada
make gpl-xmlada
make gpl-gps
```

From: Björn Persson <rombo.bjorn.persson@sverige.nu>
Subject: Re: [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.

lcov — Coverage analysis on Windows

From: Manuel Collado <m.collado@lml.ls.fi.upm.es>
Date: Thu, 07 Sep 2006 12:40:33 +0200
Subject: Re: Coverage analysis on Windows
Newsgroups: comp.lang.ada

> I am looking for a windows tool which enables to display gcov (gcc 3.4.4)

coverage results.
Would you know one?
I have tried lcov, but it isn't ported for Windows …

'lcov' is just a set of Perl scripts. They can be executed in any Windows system with a Perl interpreter (I've just tested it).

From: Manuel Collado <m.collado@lml.ls.fi.upm.es>
Date: Fri, 08 Sep 2006 13:29:41 +0200
Subject: Re: Coverage analysis on Windows
Newsgroups: comp.lang.ada

> lcov processes Unix-style paths. However, our version of gcc and gcov (gcc 3.4.4) encrypts windows-style paths into .obj, .gnco, .gda and .gcov files. Therefore, lcov does not manage them correctly …

It works on Cygwin. Cygwin is a Windows port of a large set of GNU utilities. Includes a bash shell that provides a Unix-like environment. The Perl interpreter (and any other utility) from Cygwin understand both forward and backward slashes as path delimiters.

> lcov would need to be fully ported to windows …

It works OK in my Windows XP machine via Cygwin. […]

Ada and Software Engineering Library

From: Dick Gayler <gaylers@mindspring.com>
Subject: Ada and Software Engineering Library Version 2
Date: Sun, 04 Jun 2006 18:22:55 GMT
Newsgroups: comp.lang.ada

We have been hosting Ada and Software Engineering Library Version 2 but the person who helped maintain the site is no longer affiliated with KSU. Thus, we are in the process of "pulling the plug" on the site. Is this site frequently used and is there anyone who would like to takeover hosting the site?

From: Jeffrey Creem <jeff@thecreems.com>
Date: Mon, 05 Jun 2006 21:43:30 -0400
Subject: Re: Ada and Software Engineering Library Version 2
Newsgroups: comp.lang.ada

> We have been hosting Ada and Software Engineering Library Version 2 but the person who helped maintain the site is no longer affiliated with KSU. Thus, we are in the process of "pulling the plug" on the site. Is this site frequently used and is there anyone who would like to takeover hosting the site?

I don't think it is all that frequently used but it would be a shame of anything was lost. There is probably a lot of duplicate/old files there that are already hosted elsewhere. (e.g. some old versions of GNAT).

I may try to pull it down and sort through it but I don't currently have the resources to host it.

From: Dirk Craeynest <dirk@heli.cs.kuleuven.ac.be>
Subject: Re: Ada and Software Engineering Library Version 2
Date: 17 Jun 2006 08:17:35 +0200
Organization: Ada-Belgium, c/o Dept. of Computer Science, K.U.Leuven
Summary: A mirror is available on the Ada-Belgium ftp server.
Newsgroups: comp.lang.ada

> Maybe it could become part of Adapower or Ada World? I remember Stéphane Richard saying he had lots of space.

FWIW, since a very long time Ada-Belgium has provided a complete mirror of the "Ada and Software Engineering Library Version 2 (ASE2)" on our ftp-server.

The starting point is:
<ftp://ftp.cs.kuleuven.ac.be/pub/Ada-Belgium/cdrom/index.html>

> There are probably numerous links the site. Please don't break those if it can be avoided. If someone takes over hosting, will you set up redirections (HTTP response 301)?

Anyone who has links to this library is invited to redirect to the Ada-Belgium mirror. I will keep the ASE2 directory on-line as a permanent archive of this library.

[See also same topic in AUJ 22-4 (Dec 2001). —su]

GtkAda Contributions

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: GtkAda contributions v1.2
Date: Thu, 24 Aug 2006 21:09:00 +0200
Newsgroups: comp.lang.ada

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

This new version contains two new sections:

1. A package to access style properties. Style properties are controlled using resource files. Additionally to the appearance of a widget, they can be used for internationalization purpose. String style properties are natively UTF-8.

2. xpm2gtkada, an utility for embedding images into a GtkAda application. Normally images are attached as separate files referenced in the resource file. But sometimes one could wish to be able to link some "stock" images to the executable.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: GtkAda contributions v1.3
Date: Sun, 27 Aug 2006 18:56:12 +0200
Newsgroups: comp.lang.ada

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

Added:

1. Missing windows positioning subprograms and bug fix for Set_Property on GFloat contributed by Maxim Reznik;

2. xpm2gtkada has an option to create Pixbuf embeddable images;

3. Reference-counted Ada objects as GValue.

wxAda — wxWidgets bindings

From: Lucretia <lucretia9@lycos.co.uk>
Subject: [ANNOUNCE] wxAda (pre-pre-pre-pre-release)
Date: 7 Sep 2006 12:28:34 -0700
Newsgroups: comp.lang.ada

Just thought I'd post a little note about the status of the project. I have uploaded the source to Tigris. This is not complete and I have stalled. I have recently stumbled across major blocks which I need help with. Or if somebody wants to take over leading development, I have no problem with that; my future with wxAda isn't too clear to me at the moment.

The source isn't in the best way, but it does build with wxWidgets 2.6.3 and does provide some functionality.

Please feel free to look over the project and see what can be done.

[http://wxada.tigris.org —su]

AutoIT — Automated GUI Testing

From: Per Sandberg <per.sandberg@bredband.net>
Subject: [ANNOUCE] AutoIT in Ada (Automated GUI testing) Version 0.5.1
Date: Fri, 25 Aug 2006 22:37:56 +0200
Newsgroups: comp.lang.ada

First release of auto-it in Ada, an Ada binding to the AutoIT dll that makes it possible to automate GUI tests on Windows.

Requirements:
A Win32 box
GNAT GPL 2006 / or better

URL: https://sourceforge.net/projects/ada-autoit/

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

For documentation and background.

ASIS2XML

From: Simon Wright <simon@pushface.org>
Subject: ASIS2XML 20060827 released
Date: Sun, 27 Aug 2006 18:58:30 +0100
Newsgroups: comp.lang.ada

Now released at
http://sourceforge.net/project/showfiles.php?group_id=104293&package_id=192761

There have been two silent releases since this was added to the GNAT-ASIS project on SourceForge:

20060610

Notes: This is the first SourceForge release of asis2xml.

Changes: Working on Traits so that they have a more natural interpretation: for example, An_Access_Definition_Trait on an element el becomes
`<el access="true"/>`

20060806

Notes: This release supports multi-unit programs, and includes unit name and file location.

Changes: If you've built your program using a GPR

```
$ gnatmake -Pfoo -gnatct
```

which puts the Ada library info into say .build you can process the library by

```
$ asis2xml .build >foo.xml
```

Declaration elements that correspond to units (ie, source files) include the attributes file (full source path), unit (Ada unit name).

The 'size' attributes are missing (problems with generic instantiations).

20060827

Notes: This release includes context clauses and record component sizes.

Changes: The schema has changed; now you get

```
<asis>
  <compilation_unit
file="/where/ever/foo.ads"
unit="Foo">
    <context_clauses/>
    <unit_declaration/>
    <compilation_pragmas/>
  </compilation_unit>
</asis>
```

where the contents of <unit_declaration/> are as before.

The usage has changed:

```
 usage: ./asis2xml [flags]
directory|unit.adt

 flags:  -s  report data sizes
```

For as-yet-unknown reasons, an exception can be raised if you give -s on an adt derived from a body; it seems to work fine on specs.

[See also same topic in AUJ 25-4 (Dec 2004), p.191. —su]

Avatox — Ada to XML

From: Marc A. Criley <mc@mckae.com>
Subject: Announce: Avatox 1.0 is now available
Date: Thu, 17 Aug 2006 00:58:17 GMT
Newsgroups: comp.lang.ada

Avatox 1.0 (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.

It is now available in source code form at www.mckae.com/avatox.html.

Avatox' XML representation provides some content, such as identifier names and operators, as attributes, rather than outputting everything as an element. It also outputs the row/column span of every element, and embeds comment lines within non-ASIS "A_COMMENT" elements.

Here's a brief excerpt generated by:

```
   avatox avatox.adb -Imckae
-I$GNAT/include/asis
```

(assuming GNAT is the GNAT GPL 2006 installation directory)

```
<A_DECLARATION startLine="47"
endLine="299" startCol="1"
endCol="11">
<A_PROCEDURE_BODY_DECLARATION
startLine="47" endLine="299"
startCol="1" endCol="11">
</A_PROCEDURE_BODY_DECLARATIO
N>
   <A_DEFINING_NAME
startLine="47" endLine="47"
startCol="11" endCol="16">
     <A_DEFINING_IDENTIFIER
name="Avatox" startLine="47"
endLine="47"
startCol="11" endCol="16"/>
   </A_DEFINING_NAME>
```

The canonical style of XML element naming can also be selected for ASIS element naming, which alters the previous excerpt to look like this:

```
<aDeclaration startLine="47"
endLine="299" startCol="1"
endCol="11">
 <aProcedureBodyDeclaration
startLine="47" endLine="299"
startCol="1" endCol="11">

 </aProcedureBodyDeclaration>
   <aDefiningName
startLine="47" endLine="47"
startCol="11" endCol="16">
     <aDefiningIdentifier
name="Avatox" startLine="47"
endLine="47"
startCol="11" endCol="16"/>
   </aDefiningName>
```

So what can you do with Avatox?

Well, you can now leverage XML technologies, like XPath and XQuery, for source code metrics and analysis.

There's enough information in the generated XML representation that one should be able to identically reconstitute the original Ada source code. And should such a utility come into existence <grin>, one could even do things like XSLT based transformations on the XML and generate Ada source code from the resulting file.

Avatox 1.0 was developed using GNAT GPL 2006, and is an evolution of the Display_Source program distributed with that compiler's ASIS distribution. Avatox is therefore licensed under the GPL, while the included McKae software utilities is licensed as GMGPL.

Check it out at www.mckae.com/avatox.html. (And yes, I know the title graphic is missing, it's on order :-)

From: Marc A. Criley <mc@mckae.com>
Subject: Announce: Avatox 1.1 now available
Date: Sun, 20 Aug 2006 14:48:13 GMT
Newsgroups: comp.lang.ada

(Though it's been scarcely a week since the 1.0 release 1.1 is already on the net. Hopefully there won't be too many applications dependent on the 1.0 Avatox XML Format that will be impacted by this update :-)

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.

With 1.1, that ASIS representation is now wrapped in a containing element that also includes information about the elements being used to express the representation of the source code. The intent of these element "pedigrees" is to support the extension of the Avatox XML Format (AXF) and simplify the processing performed by any follow-on tools.

Avatox 1.1 is available at the McKae Technologies website at www.mckae.com/avatox.html.

From: Marc A. Criley <mc@mckae.com>
Subject: Re: Announce: Avatox 1.0 is now available
In-Reply-To: <m2odudyfdm.fsf@grendel.local>
Date: Thu, 24 Aug 2006 00:41:18 GMT
Newsgroups: comp.lang.ada

Simon Wright wrote:

> I took the view that <A_DEFINING_NAME/> would look better as <defining_name/> throughout.

I try to adopt the prevailing style and idioms of a given programming or similar such language, hence the camelBack no underscore approach.

> There are also some interesting choices about which aspects to map into attributes and which into child elements, and indeed on the whole mapping to be used.

Basically, I mapped literals into attributes, like identifiers, operators, numbers, and such.

There are other possibilities, but I didn't want to have to maintain any kind of info stack to know what value to associate with a previous or subsequent element. Like A_CLAUSE could've had the kind

of clause--"with", "usePackage"--as an attribute. Here it's all done in a single pass with no look back or look-ahead.

> Do you support multi-unit environments?

The current release of Avatox only processes the single compilation unit provided to it. Extending that to a closure probably wouldn't be that hard.

The Avatox XML Format (AXF) would support multiple units as-is (v1.1 and beyond), you just keep adding them in.

> Do you think there'd be a future in offering at least the concept to the ASIS team?

I'm a little unclear on what concept you're referring to…an XML definition for Ada?

> As a demo of the sort of thing one can do, we found it pretty straightforward to generate a report …

The analysis and reporting aspect is definitely one area for Avatox, using stylesheets just as you've done.

My focus though is more on "Transformation & Vivification", i.e., transforming the XML representation using <buzzword>XML-enabled technologies</buzzword> and then bringing the result to life by some means, whether that be converting the XML back to Ada source code, or compiling it directly, or through some other animation.

I'm experimenting and watching to see where this goes…

GADBTK — Generic Avionics Data Bus Tool Kit

From: okellogg <okellogg@freenet.de>
Subject: GADBTK ATIP1553 GNAT?
Date: 31 Aug 2006 07:43:09 -0700
Newsgroups: comp.lang.ada

Has anybody ported the The Generic Avionics Data Bus Tool Kit (GADBTK) MIL-STD-1553 Ada binding to GNAT? The version at AdaIC is for DEC Ada:

http://archive.adaic.com/tools/bindings/gadbtk/

> "atip1553.tgz,182 KBytes) The Generic Avionics Data Bus Tool Kit (GADBTK) provides a strong Ada software binding to the military standard 1553 data bus. The bus is used for time multiplex data communications between different sensor and computer subsystems on many current military platforms. [NR]"

Qt4Ada — Qt 4 Binding

From: Yves Bailly <kafka.fr@laposte.net>
Subject: Qt4Ada: Qt for Ada
Date: Sat, 29 Jul 2006 12:34:04 +0200
Newsgroups: comp.lang.ada

After reading the thread about an Ada 2005 binding for the Qt library, it seems obvious that many would be interested in such a work.

So, let me announce I have started such a binding some times ago, a binding to Qt version 4.1.4.

It's only the beginning, however it's already usable: the 6th Qt tutorial (http://doc.trolltech.com/4.1/tutorial-t6.html) has been successfully reimplemented in pure Ada.

The binding is done through a intermediate C interface, using GNAT 2006 as compiler. However I'm trying to write things as portable as possible, not relying on compiler-specific features. The actual binding is a rather thick one, the thin binding being limited to the C interface.

I pay particular attention to reflect in Ada the type structure from C++. At the top there is one package, named "Qt". Below there are child packages, corresponding to Qt's modules, for now only "Qt.Core" and "Qt.Gui". Then each class has its own package, for example "Qt.Core.QStrings" for QString class, "Qt.Gui.QWidgets" for QWidget class. The hierarchy of types is also preserved: the type Qt.Gui.QWidgets.QWidget is derived from the type Qt.Core.QObjects.QObject, and so on.

Please note I'm not an Ada expert — but I'm willing to learn and get better. So even a quick review by someone more skilled would be very appreciated ;-)

Now some numbers.

There are more than 5000 functions or methods in Qt4. With the current structure, when not asleep, it needs roughly 1min30sec to bind a C++ method to an Ada function or procedure, in the simplest cases. So it would need at least 125 hours to bind everything, more than 5 days if working 24 hours a day, or about 16 days if working 8 hours a day. Applying the usual correction factor for projects planning, we get about 50 days of full-time work. Which is not that much, after all.

But I'm not working full-time on this project. It's a "free" project, done on my spare time. […]

So, by myself I can only work on Qt4Ada something like 1 (one) hour a day (it's a weekly mean, some days I work more, some days not at all). At this pace, the binding would be achieved in more than one year, assuming everything goes well. No, I'm not using some automatic translator: as someone already told, for such complex things as Gtk or Qt such translators would in fine do more harm than good. […]

[See also "QtAda binding" in AUJ 26-4 (Dec 2005), pp.237–238. —su]

From: Michael Bode <m.g.bode@web.de>
Subject: Re: Qt4Ada: Qt for Ada
Date: Sat, 29 Jul 2006 14:35:35 +0200
Newsgroups: comp.lang.ada

> On the "administrative" side, I have some wonder about licensing. Has it's only a personal project, I think GPLv2 would be fine. What's your opinion?

Since you started the Qt4Ada thread with the words:

I just read the thread "Answer of Request to AdaCore on licensing Status of GtkAda 2.4.0",

I'd suggest using GMGPL. But this depends on what you want to achieve with Qt4Ada:

Do you want to promote the Free Software idea in the first place? Then go with the GPL.

Do you want to create a library that is useful to the widest range of programming projects (Free and Non-Free)? Then use GMGPL.

Do you want to to create an alternative to GtkAda because you think it sucks technically or you like Qt better? Then both are viable.

Do you want to create an alternative to GtkAda because of the licensing mess there? Then go with GMGPL.

Of course since Qt itself uses a dual licensing scheme where you have to pay if you want to do CSS development (but not as astronomical as for GtkAda), you could invent a similar dual license scheme and maybe earn some money.

Anyway, please make it crystal clear what your licensing is and stay with it.

From: Jeffrey Creem <jeff@thecreems.com>
Subject: Re: Qt4Ada: Qt for Ada
Date: Sat, 29 Jul 2006 13:44:28 -0400
Newsgroups: comp.lang.ada

> Does it matter if QT4Ada is GMGPL when QT is GPL? I mean shouldn't QT4Ada then be dual licensed so that whether one use the QT GPL or non-GPL version one do not get problems?

GMGPL is "GPL Compatible", so GMGPL can certainly be the only way this library is licensed (assuming that meets the needs of the developers that sign up).

So, I don't see a reason for dual license if GMGPL is what is selected.

From: Michael Bode <m.g.bode@web.de>
Subject: Re: Qt4Ada: Qt for Ada
Date: Sat, 29 Jul 2006 23:50:05 +0200
Organization: 1&1 Internet AG
Newsgroups: comp.lang.ada

[…] If Qt4Ada is GPL, you can't use it for CSS even if you buy a Qt license.

For this reason I'd prefer something that is GTK+ based. And I consider GtkAda under GPL like the tail wagging the dog. After all GTK+ does the real work and it is LGPL. Now comes one of a dozen or so

language bindings and shows people how to do Free Software.

From: Yves Bailly <kafka.fr@laposte.net>
Subject: Re: Qt4Ada: Qt for Ada
Date: Tue, 01 Aug 2006 23:55:19 +0200
Newsgroups: comp.lang.ada

> Are you aware of this project at http://www.websamba.com/guibuilder ? I haven't looked at it in detail; it seems to be GPL.

Yes, I've looked at it. For one thing, it's for Qt 3.3 : my proposal targets Qt 4 — and only Qt4, so both are complementary. Second, it's a binding for Ada 95, I'm targetting Ada 2005.

> Me on my side, I think that if you do not release your binding under the GMGPL, you inhibit any (closed source) commercial use of your binding. Because closed source software can't use your binding for writing commercial programs released under GPL, whereas they could if you'd release it under the GMGPL.

Note that the GPL doesn't prohibit *commercial* software, it "only" prohibits *closed source* software. I guess we could argue on this till the end of times, but I do believe that "commercial" /= "closed source". And I also believe (again an arguable opinion) that open source is better to produce good software. For now I'm producing closed source software, for which I'm paid so I can live. Despite the fact it pleases some consumers, who don't want to pay too much, it's badly written, and it's the case of almost all closed source software I've had the opportunity to read the source code. On the hand, most open source software are better written. At least, this is my own experience until now, I won't say it will never change in the future.

> I share Jeffry's concern that you may very well be unable to switch back to GMGPL later on, once the community has started sending source updates on your binding.

I understand this. But given the previous opinions, it's not really a concern for me.

I've reached the conclusion that the "best" model would be some kind of dual-licensing, something like TrollTech does for Qt. However I don't see, at least for now, how I could "enforce" a commercial license, charge fees for it, and so on (not to mention that this project is still at its beginning and not yet really usable). Dual licensing implies many legal and administrative stuffs, which I'm not ready nor able to handle by myself. Again, this might change. I have to discuss of all this with a lawyer I know and with my current boss. Who knows…

So, to sum things up: for now, I tend to prefer the GPL. But if something new happens soon (some legal knowledge I don't have, some administrative support…), I might go for a dual license.

From: Michael Bode <michael.bode@laserline.de>
Subject: Re: Poll: Qt4Ada as alternative to GtkAda
Date: 28 Jul 2006 12:24:39 +0200
Newsgroups: comp.lang.ada

> But the real issue isn't if the GtkAda is GPL or not as long as the GNAT compiler only can be used to make GPL code then it really doesn't matter what license your library has.

But I think this is quite clear: gnat from FSF still has the linking exception. GCC-GNAT 4.1 is available for some Linux distributions (Mac OS X?) and I think MinGW gnat 3.4.5 is available for Windows.

So the problem are some libraries the most important being some decent multiplatform GUI lib.

From: Simon Clubley <clubley@eisner.decus.org>
Subject: Re: Poll: Qt4Ada as alternative to GtkAda
Date: 28 Jul 2006 08:17:52 -0500
Newsgroups: comp.lang.ada

> So it means that what AdaCore contributes to FSF (GCC) is GMGPL, while what they package themselves is GPL?

Yes, that's right. If you pull a FSF GCC distribution, with a FSF version number, from a FSF server, it's my understanding that the Ada RTL component is licensed under the GMGPL.

A theoretical concern that I had a few weeks ago was could ACT, at a later date, move the GNAT.* packages in the FSF distribution to been GPL only on the basis that they were not part of the Ada 95 standard, and hence, like GtkAda, ACT was free to do with them whatever they wanted to do?

From: Jeffrey Creem <jeff@thecreems.com>
Date: Fri, 28 Jul 2006 10:08:41 -0400
Subject: Re: Poll: Qt4Ada as alternative to GtkAda
Newsgroups: comp.lang.ada

I would expect some discussion on the GCC group before such a move happens. For items in the actual FSF GCC tree, ACT has to assign copyright to the FSF. So, while they are always free to stop contributing, I don't think AdaCore by themselves can change the license terms on items pulled from the FSF tree.

Of course, the FSF could make a change like that. While AdaCore's actions are (hopefully) driven by profit motives, FSF's motives are simply trying to ensure an end state where software is "Free" (in a GPL sense). So they could certainly change future releases to pure GPL for their own reasons (and of course a proprietary vendor could change future license terms to require something unacceptable in future versions too). At least with open source and either very shallow pockets (lawsuit proof) or fairly deep pockets (lawyer up), one could branch from the last set of acceptable license terms…

In any case, on the original point of this thread, it is hard to understand why we would abandon GtkAda for Qt4Ada when Qt itself is GPL without exception on some platforms.

http://www.trolltech.com/developer/downloads/qt/windows

So, how would the community be any better off? Yes you can buy a commercial license for it.. But of course you can by GtkAda from AdaCore too.

Now, having an alternate or additional GUI library support is not a bad thing (perhaps it is even a good thing) but I don't think QT really solves the license problems that most people are worrying about.

ABDI — Ada Unified Database Interface

From: Maxim Reznik <reznikmm@front.ru>
Subject: Re: Custom model in gtkada?
Date: Wed, 14 Jun 2006 21:00:37
Newsgroups: comp.lang.ada

ABDI is an unified database interface for Ada. Now it works with Firebird and Oracle. You can get it http://www.ada-ru.org/files/adbi-0.2.tar.bz2

Vim Ada-Mode

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: Ada-Mode for vim updated.
Date: Sun, 16 Jul 2006 18:01:49 +0200
Newsgroups: comp.lang.ada

If you are using Ada and VIM then I have something for you: A complete refurbished Ada-Mode for vim.

http://www.vim.org/scripts/script.php?script_id=1609

The new mode is now distributed as Vim-Ball for easy installation and features its own on-line help (just type ":help ada.txt") for easy setup and usage.

Tag search has been updated using vim new ability to set the quick-fix list under script control. A lot more helpful then the ugly tjump list.

But the coolest extra is omni-completion which allows syntax completion across the whole project and not just the file(s) loaded. Provided you have a "tags." file with all the identifiers in it.

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Thu, 10 Aug 2006 20:31:50 +0300
Subject: VIM Ada-Mode 3.5

Newsgroups: comp.lang.ada

The Vim Ada-Mode 3.5 [1] is out and it has some really interesting features which make development interesting. You can have a look at what it can do here:

http://sourceforge.net/project/screenshots.php?group_id=3D12974

The Ada-Mode mode supports GNAT and Dec Ada and Vim runs on quite a few more platforms than the GPS.

[1] http://www.vim.org/scripts/script.php?script_id=3D1609

Convert_prj — GPR to ADP

From: M E Leypold <kontakt@m-e-leypold.de>
Subject: Re: Meet the new Date: 28 Jun 2006 01:06:35 +0200
GPS…same as the old GPS…
Newsgroups: comp.lang.ada

> I would like to use Emacs as the "IDE" as I do for other languages. Are you using the Ada-mode supplied with Emacs 21.x? Or are you using the Ada-mode from ACT, or some other Ada-mode?

I use the one coming with Debian. Whichever version that is.

> I don't understand how to "install" the Ada-mode files (about 5 Emacs lisp sources) that I downloaded from the Libre site. As I mentioned above, the Ada-mode that comes with Emacs has a significant problem: when you select Build from the Ada menu in Emacs, it builds the source > you're looking at.

You need to create a .adp file to set the "main file" to build. Then it builds always your program. Use Ada→Project→New from the Emacs menubar to create a new project file for Ada mode.

> And then, if you try to build again while in a new source (even after killing the buffer for the previous source) it builds the old source again.

Since it constructed a default adp file from the first file.

> I have not found a way, other than killing that instance of Emacs and starting a new one, to ever build more than one source module. That doesn't sound like a good productivity proposition!

No. I missed the point for some time too. Since then I generate (that is a bit unusual) a adp file for every executable from the Makefile.

```
statfix.adb  -> statfix.adp
createdb.adb -> createdb.adp
```

etc.

Now on first compilation I'm asked what I want to compile, except if the file that is open is one of the main files, then the selection is automatically.

The debugger is a pain in Emacs, though. It's workable, but you have to convince yourself.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Meet the new GPS…same as the old GPS…
Date: 28 Jun 2006 02:23:08 -0700
Newsgroups: comp.lang.ada

Both versions are available [in Debian]; one bundled with Emacs, and the more advanced one in the "ada-mode" package.

Personally I use a modified ada-mode that does not use .adp files at all, it uses the .gpr files directly. There are limitations, but it does mostly what I need. I'll make my changes availale somewhere on request.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Meet the new GPS…same as the old GPS…
Date: Wed, 28 Jun 2006 23:07:26 +0200
Newsgroups: comp.lang.ada

> I don't know if you know the AdaCore GLIDE? it uses a special program convert_prj which converts a GPR to a nearly-equivalent ADP.
> I see that the sources of convert_prj are (c) FSF …

Yes, I'm aware of it. convert_prj is also in the sources of GPS, but I haven't taken the time to look at it. I'm reluctant however to use ..adp files at all, even generated, when I could use only the .gpr files.

Emacs Ada-Mode

From: Georg Bauhaus <bauhaus@futureapps.de>
Subject: Re: looking for emcas ada2005 mode
Date: Tue, 27 Jun 2006 11:51:32 +0200
Newsgroups: comp.lang.ada

> Does anyone known where it is? Or still in progress?

According to one of the authors of Ada mode for Emacs, who works for the producers of GPS, they are concentrating on GPS ("we never looked back").

Some of the Emacs Lisp can be changed easily, like adding the three new keywords to the list of reserved words.

From: Stephen Leake <Stephe.Leake@nasa.gov>
Subject: Re: looking for emcas ada2005 mode
Date: Thu, 29 Jun 2006 13:31:00 -0400
Newsgroups: comp.lang.ada

I have volunteered to be the new Gnu Emacs Ada mode maintainer.

However, I am waiting for AdaCore to release their test suite for the indentation engine. They have some proprietary (customer) Ada code in the test suite, so they have to strip that out.

If you could pressure them to do that, things would move forward.

I've added the new Ada 2005 keywords to my copy of Emacs Ada mode, but only for fontify, not for indentation.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: New mainainer for emcas ada2005 mode
Date: Fri, 30 Jun 2006 23:53:06 +0200
Newsgroups: comp.lang.ada

> Would you like to join the GNU Ada Project? We already maintain the current Ada mode for vim [2] and would welcome Emacs as well.
> Just get youself an sourceforce user and drop us a support request [3].

Please consider joining the upstream Emacs project instead.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: New mainainer for emcas ada2005 mode
Date: Sun, 02 Jul 2006 19:19:48 +0200
Newsgroups: comp.lang.ada

> Sure. But joining the upstream is also an upstream struggle. Ada is then only one minor part of the whole project. For example: In the vim development the runtime files/scripts are only changed when a new version comes out. vim 7.0 is fairly new so it can take quite a while until vim 7.1 comes out.

That is a short-sighted argument. Yes, it takes more effort to join the upstream team. So what? The benefits are well worth it. As more and more tools gain native support for Ada more people will be intrigued and consider using Ada in their software. The upstream authors themselves might be sufficiently intrigued to investigate the language. Contrast this with separate add-ons that require potential users:

- to be aware of their existence
- to be convinced that they are well integrated in their tool

- to actively look for them

- to download them

- to install them

- to maintain them when their tool changes

That is the reason why I have joined the Debian GCC maintainers, rather than maintaining gnat separately from GCC. And that is the reason why, a while ago, I recommended that you do the same for your distribution of choice.

From: Stephen Leake <Stephe.Leake@nasa.gov>
Subject: Re: New mainainer for emcas ada2005 mode
Date: Tue, 04 Jul 2006 09:12:22 -0400
Newsgroups: comp.lang.ada

> And that is also why GNAT was integrated into GCC, raising awareness of Ada to unprecedented levels.

I plan to join the 'upstream Emacs' project. However, it may also be useful, as Martin points out, to have a place to do intermediate releases. The Sourceforge Gnu Ada project seems like a reasonable place, although a Savannah project might be more appropriate.

I hope to set up a Gnu Emacs Ada mode mailing list somewhere, to discuss the future of Ada mode, and details about proposed changes.

Ada refactoring tools

From: trg <trg@world. std.com>
Subject: Ada refactoring tools?
Date: Tue, 8 Aug 2006 18:59:35 +0200
Newsgroups: comp.lang.ada

I'm looking for tools that will do refactoring for Ada 95+. I've seen Adasubst/Adadep. What else is available?

Other than dedicated tools, what sort of refactoring features exist in current Ada environments (beyond gnatpp)?

Any suggestions on refactoring features for Ada that you haven't seen but would find useful?

From: pth@darrach.net
Subject: Re: Ada refactoring tools?
Date: 10 Aug 2006 11:53:38 -0700
Newsgroups: comp.lang.ada

You can check out Headway Review, at http://headwaysoftware.com/products/review/ada/.

It does "prefactoring" i.e. the analysis of the changes you might want to make.

But do bear in mind that Review is in the process of being replaced by Structure101, which is currently only available for Java. Ada support is expected to see the light of day before the end of the year.

Review is a good place to start.

Ada Plugin for Eclipse

From: Fifth Horseman <asmjk1@comcast.net>
Subject: Eclipse Plug-In
Date: Thu, 10 Aug 2006 10:57:25 -0600
Newsgroups: comp.lang.ada

Time to revive the Ada plug-in for Eclipse discussion. I haven't seen any posts more recent than a year ago, so I am about to get started on trying to create my own solution.

If anyone has a working (or partially-working) solution they're willing to share, I'd love to see it. I don't have any prior experience writing Eclipse plug-ins, so I might be getting in over my head, but we'll see how successful I am.

From: Jeffrey Creem <jeff@thecreems.com>
Subject: Re: Eclipse Plug-In
Date: Thu, 10 Aug 2006 22:15:39 -0400
Newsgroups: comp.lang.ada

The CDT group (C/C++ developers toolkit) seemed interested and eager to help anyone interested in extending that toolkit to support Ada though there really has not been any mail list traffic about that in about a year.

It is not clear to me that adding on top of CDT is the quickest/best choice. It may be faster to do something more direct and limited but it is worth a look.

Other are correct that there exists non-publically available eclipse plugins. (Aonix and DDC-I and to some extent AdaCore).

Another thought for a starting point is that there are several colorizer plugins for eclipse that are Ada aware. That is obvious a far cry from real Eclipse integration.

From: Stephen Leake <stephen_leake@acm.oSubject: Re: Eclipse Plug-In rg>
Date: Sat, 12 Aug 2006 08:10:52 -0400
Newsgroups: comp.lang.ada

AdaCore just announced GNATbench, a plug-in for Eclipse fully supporting GNAT.

It will probably make it into a public release sometime.

[http://www.adacore.com/home/gnatpro/toolsuite/gnatbench —su]

From: Tom Grosman, Aonix <grosman@aonix.fr>
Subject: Re: Eclipse Plug-In
Date: Wed, 16 Aug 2006 19:41:44 +0200
Organization: Aonix
Newsgroups: comp.lang.ada

> Well, Aonix made one and in the small print they say it even supports GNAT.

Yes, AonixADT (Ada Development Toolkit) is an Ada development plug-in for Eclipse. It supports ObjectAda as well as GNAT (and not just in small print :-). ADT currently runs under MS Windows, Intel Linux and Sparc Solaris environments.

We will very shortly be releasing the latest version which will be freely available, so stay tuned.

Tom Grosman, Aonix

[http://www.aonix.com/adt.html --su]

GHDL — VHDL simulator

From: Tristan Gingold <tgingold@REM.free.fr>
Subject: GHDL 0.25 is released
Date: 14 Aug 2006 11:53:31 GMT
Newsgroups: comp.lang.ada

I have just released GHDL 0.25. GHDL is a complete VHDL simulator. You can download it or get more info from http://ghdl.free.fr

I post this announce on CLA for three reasons:

* GHDL is written in Ada

* On Linux, GHDL is a GCC front-end (like GNAT, it doesn't generate C code).

* On Windows, GHDL directly generate x86 code. As far as I know, GHDL is the only OSS/Free Ada program which includes a JIT.

Ada-related Products

AdaCore — GNATbench Eclipse Plug-in

URL: http://www.adacore.com/2006/06/19/new-plug-in-bridges-the-gap-between-gnat-pro-and-eclipse/

Monday June 19, 2006

New AdaCore Plug-in Bridges the Gap Between GNAT Pro and Eclipse

NEW YORK, NY, USA — AdaCore today announced its latest technology addition — a stand-alone version of the company's GNATbench plug-in that integrates Ada into native (standard) Eclipse. GNATbench offers the familiar Eclipse "look and feel" when utilizing GNAT Pro's leading-edge compiler, tools and capabilities, and supports both all-Ada and mixed-language development. With one simple plug-in, GNATbench lets Eclipse users access the benefits of GNAT Pro Ada to develop more reliable applications with fast, predictable performance and at lower cost.

GNATbench was originally developed for Wind River's Eclipse-based Workbench development suite to facilitate multi-language development, sophisticated Ada-aware editing, code browsing, debugging, comprehensive compilation, as well as prototyping and simulation for advanced VxWorks systems creation. The new GNATbench configuration for Eclipse is a separate plug-in that offers all the editing and browsing features of the Workbench version, including the Outline View — a high-level view of the code to facilitate program comprehension and development. The difference is the intended execution target: the builder produces executables for native systems, rather than embedded processors, and likewise the debugger supports native system debugging. GNATbench for Eclipse is currently supported on Windows and Solaris host environments.

"The growing popularity of the Eclipse framework in embedded application development motivated us to add a native Eclipse plug-in to our comprehensive GNAT Pro Ada toolsuite," said Robert Dewar, President of AdaCore. "Our new plug-in not only integrates Ada seamlessly into Eclipse, it also provides assurance to our Ada customers that they

always have access to the latest tools, language and toolset experts to support their multi-language embedded application development projects."

About GNAT Pro

GNATbench draws its strength from AdaCore's GNAT Pro, a robust, flexible, and open Ada development environment with user-friendly licensing, based on the GNU GCC compiler technology. It comprises a full Ada compiler (including support for all major Ada 2005 features), an Integrated Development Environment (GPS, the GNAT Programming Studio), a comprehensive toolset including a visual debugger, and a useful collection of libraries / bindings. GNAT Pro allows development of pure Ada applications as well as Ada components in multi-language systems. It is distributed with complete source code, and is backed by rapid and expert support service. GPS is available on a wide range of host environments for both native and cross-development using GNAT Pro, including Unix, Windows and GNU/Linux. As evidenced by its successful application by customers worldwide, GNAT Pro is the best choice for reliable and efficient software, across a wide spectrum of applications, including high-integrity systems.

Pricing and Availability

The GNATbench plug-in for Eclipse will be available starting July 2006 as part of the GNAT Pro subscription. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations.

About AdaCore

Founded in 1994, AdaCore is the leading provider of commercial, open-source software solutions for Ada, a modern programming language designed for large, long-lived applications where reliability, efficiency and safety are absolutely critical. AdaCore's flagship product is GNAT Pro, the commercial-grade open-source Ada development environment, 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 defense 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 — GNAT Pro Preview release

From: Romain Berrendonner <berrendo@adacore.com>
To: announce@adacore.com
Subject: [AdaCore] Announcing beta program for GNAT Pro new compiler back-end technology
Date: Fri, 07 Jul 2006 18:51:06 +0200
Organization: AdaCore

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.

At this stage, we can provide beta versions for sparc-solaris, x86-linux and x86-windows, and would appreciate feedback on this new technology from interested GNAT Pro users. AdaCore will provide support for beta

Note that this beta version is based on our current development version, 5.05w. It is unrelated to the forthcoming 5.04a1 release which will be announced in the coming weeks.

If you are interested in participating in this beta program, please download the packages from the Download GNAT Pro section of GNAT Tracker (www.adacore.com) and send your general feedback via gnattracker as usual.

Please don't hesitate to contact us if you have any questions on this program.

AdaCore — GNAT Programming Studio 4.0

AdaCore Delivers Enhanced Ada Integrated Development Environment
Publication date on this website: Monday, July 10, 2006
Company: AdaCore
Category: Press Releases : Tools

Summary:

AdaCore launches GNAT Programming Studio (GPS) 4.0, an advanced, powerful Integrated Development Environment (IDE) that accompanies the GNAT Pro Ada language development toolset.

Full Text:

NEW YORK, July 10, 2006 — AdaCore today launched GNAT Programming Studio (GPS) 4.0, an advanced, powerful Integrated Development Environment (IDE) that accompanies the GNAT Pro Ada language development toolset.

The completely updated version of GPS improves productivity through new features such as on-the-fly code completion, remote programming, and improved version control. The sophisticated code completion engine understands the details of Ada language semantics, enabling automatic program completion, maximizing efficiency, and cutting development time. The new remote programming function allows developers to take advantage of the power and graphical capabilities of their PCs, enabling them to work without incurring bottlenecks on servers. Version control is now tightly integrated within GPS, enabling better management of large and complex projects, particularly across sizable development teams.

Additionally, the user interface has been extensively updated to provide a modern look and feel that ensures that GPS remains easy to use by Ada programmers. Extended support for new platforms such as x86-64 Linux and PPC AIX further increases the range and usability of GPS 4.0.

"The combination of advanced programming features and its updated interface mean that GPS 4.0 leads the field in Ada development," said Arnaud Charlet, GPS Project Manager at AdaCore. "We have listened extensively to our customers and provided the most powerful environment yet for professional Ada programmers."

"The increasing use of Ada in major mission-critical software projects around the world demonstrates the need for an advanced development environment," added Robert Dewar, AdaCore's President and CEO. "The latest version of GNAT Programming Studio delivers further productivity improvements that make Ada development fast and seamless."

GPS offers advanced features such as multi-language support (including Ada, C, and C++) and is available on a wide range of host environments for both native and cross-development, including Unix, Windows and GNU/Linux. An intuitive, unified visual interface, identical across all platforms, serves as a control panel to access tools from AdaCore´s GNAT Pro Ada development environment as well as from third parties, easing both development and maintenance. As a result, GPS is particularly suited for large, complex systems requiring tool chain integration, ease of use, user customization, and code navigation/analysis.

GPS 4.0 provides many new improvements, including:

· On-the-fly code completion

· Remote programming

· Version Control improvements

· Support for branches

· File status cached between sessions

· Support for .cvsignore

· Improved user interface

· Support for x86-64 Linux and PPC AIX

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

About GPS

GPS is a powerful Integrated Development Environment (IDE) written in Ada, based on the GtkAda toolkit. GPS´ extensive source-code navigation and analysis tools can generate a broad range of useful information, including call graphs, source dependencies, project organization, and complexity metrics. It also provides support for configuration management through an interface to third-party Version Control Systems, and supports a variety of platforms, including Alpha Tru64, Altix Linux, IA64 HP Linux, IA64 HP-UX, MIPS-IRIX, PA-RISC HP-UX, PPC AIX, PPC Mac OS, SPARC Solaris, x86-64 Linux, x86 GNU/Linux, x86 Solaris, and x86 Windows. GPS is highly extensible; a simple scripting approach enables additional tool integration. It is also customizable, allowing programmers to specialize various aspects of the program´s appearance in the editor for a user-specified look and feel.

Pricing and Availability

GPS 4.0 is now available to GNAT Pro customers on selected platforms. GPS is included with the GNAT Pro Ada Development Environment. Pricing for GNAT Pro subscriptions starts at $14,000. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations.

AdaCore — GNAT Pro 5.04a1

From: Romain Berrendonner <berrendo@adacore.com>
To: announce@adacore.com
Subject: [AdaCore] F605-008 Announcing the availability of GNAT Pro 5.04a1
Date: Fri, 28 Jul 2006 18:17:00 +0200
Organization: AdaCore

AdaCore is pleased to announce the immediate availability of the GNAT Pro 5.04a1 release for the following native platforms:

alpha-openvms
alpha-tru64
pa-hpux
ppc-aix
ppc-darwin
sparc-solaris
x86-solaris
x86-linux
x86-windows
x86_64-linux
ia64-hp_linux
ia64-hpux
ia64-sgi_linux

and the following cross platforms:

ppc-vxw-windows
ppc-vxw-solaris
ppc-vx178b-windows
ppc-vx178b-solaris
ppc-vx653-windows

The distributions can be downloaded as usual using GNAT Tracker. Note that, for your convenience, GNAT Tracker can now be accessed directly from the AdaCore home page (http://www.adacore.com).

GNAT Pro 5.04a1 provides fixes for issues reported in the 5.04a release and documented in the known-problems-504a file (available through GNAT Tracker).

The 5.04a1 release also comes with an improved version of our GPS IDE, GPS 4.0. This major update provides many new improvements, including:

· On-the-fly code completion
· Remote programming
· Version Control improvements
  - Support for branches
  - File status cached between sessions
  - Support for .cvsignore
· Improved user interface
· Support for x86-64 Linux

We encourage you to install and start using this latest version of the GNAT Pro tool suite. As always, for questions, or to inform us of issues that you encounter, please let us know through the GNAT Tracker report facility or by email at the usual report@adacore.com address.

You may also be pleased to learn that we have created a section on our website dedicated to technical information surrounding GNAT Pro and Ada. The Developers Center includes a developers log giving updates on GNAT Pro technology, technical papers, code samples, and documentation. For more info, please visit:

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

[See also “AdaCore — GNAT Pro 5.04a” in AUJ 27-2 (Jun 2006), pp.73–74. —su]

AdaCore — ZCX for VxWorks

Zero cost exceptions supported for VxWorks

Tuesday July 4, 2006

As of the 5.04a1 release, zero cost exceptions are available on VxWorks 5.x, and an appropriate ZCX run time is included with the release. An earlier version of this support was available with the 5.03 release, and now with 5.04a1, this is fully supported. Note that ZCX is not yet supported on VxWorks 6.

AdaCore — GNAT Pro for Alpha OpenVMS 8.2

AdaCore extends GNAT Pro 5.04a1 support to Alpha OpenVMS 8.2

Friday September 8, 2006

AdaCore is pleased to announce that GNAT Pro 5.04a1 now supports Alpha OpenVMS 8.2. GNAT Pro has been ported to more platforms, both native and embedded, than any other Ada technology. For a full list of supported configurations, please visit our configurations page

Aonix — ObjectAda for Windows 8.2

From: Owner-Intel-ObjectAda <owner-intel-objectada@aonix.com>
Subject: Intel-OA: New ObjectAda 8.2 Update
Date: Fri, 15 Sep 2006 14:49:16 -0700
To: intel-objectada@aonix.com

A new update for Aonix ObjectAda for Windows 8.2, 1102V82-U3, is now available at http://www.aonix.com/ada_patches.html.

Please see the Release Notes for further details on the corrections made and installation instructions. The release notes can be viewed at ftp://ftp.aonix.com/pub/adats/outgoing/1102/8.2/U3/1102V82-U3.Release_Notes.

Downloading ObjectAda updates requires a password which can be obtained from your local Aonix Customer Support department.

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 “Aonix — ObjectAda Update” in AUJ 27-2 (Jun 2006), p.76. —su]

DDC-I — Wind River Workbench and SCORE

DDC-I's SCORE Compilers Available to Wind River Workbench Developers Targeting VxWorks 6.3

Safety-critical Ada, C, and EC++ applications developed using SCORE for Workbench can now run in real time under VxWorks

Phoenix, AZ. July 26, 2006. DDC-I, a leading supplier of development tools for safety-critical applications, today announced the availability of its SCORE® compilers for the Wind River Workbench, an eclipse-based development suite, and Wind River's VxWorks real-time operating system. The integration enables developers working within a Wind River Workbench environment to utilize SCORE tools to develop mixed Ada, C, and Embedded C++ applications for deployment on VxWorks target systems.

"The SCORE toolset addresses all aspects of safety-critical application development, debugging, testing, and deployment on VxWorks target systems," said Bob Morris, president and CEO of DDC-I.

"Now, developers can utilize the SCORE tools within a Wind River Workbench environment to develop mixed language applications, with full access to advanced tools such as dynamic visual process, task, and thread debugging, system analysis and validation."

"Wind River is committed to delivering a standards based development suite that provides multi-language support in a single development environment," said Steve LaPedis, vice president of strategic alliances, Wind River. "With DDC-I's SCORE tools tightly integrated with Wind River's Workbench development suite, C, C++ and Ada development can all be supported in a single environment. This integration enables developers to create reliable, optimized code for a broad range of safety-critical applications targeting VxWorks systems."

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. To support VxWorks, DDC-I has mapped its own bare run-time system to VxWorks, including all system calls, multitasking, and interrupt processing facilities. In this implementation, SCORE kernel calls are mapped to VxWorks calls, and Ada tasks are mapped to VxWorks tasks.

SCORE is available immediately for Workbench and VxWorks. 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. For more information regarding DDC-I products, contact DDC-I at 1825 E. Northern Ave., Suite #125, Phoenix, Arizona 85020; phone (602) 275-7172; fax (602) 252-6054; e-mail sales@ddci.com or visit www.ddci.com.

DDC-I — Enhanced SCORE IDE

DDC-I Announces Enhanced SCORE Integrated Development Environment for Mixed Ada, C, and EC++ Applications

Features enhanced compiler/run-time safety for multithreading, RTX and VxWorks run-time support, Workbench integration, and support for the latest Vector test tools

Phoenix, AZ. July 18, 2006. DDC-I, a leading supplier of development tools for safety-critical applications, today announced version 2.7 of its SCORE® Integrated Development Environment (IDE). The new IDE features enhanced safety for compiler and run-time multi-threading, run-time support for the VxWorks real-time OS and RTX Windows real-time extensions, and seamless integration with Wind River's Workbench IDE. SCORE version 2.7 also provides support for the latest version of Vector Software's VectorCAST test tools.

"The SCORE IDE addresses all aspects of safety-critical application development, debugging, testing, and deployment on the target system," said Bob Morris, president and CEO of DDC-I. "Version 2.7 takes SCORE to the next level by giving SCORE developers access to advanced tools."

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, tasking, exceptions, break-on-exception and break-on-tasking events. The debugger is non intrusive, can debug at the source or machine level, and can be enabled without changing the generated code.

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. SCORE version 2.7 adds two new run-time options: Wind River's VxWorks GPP 6.3 (General Purpose Platform) and Ardence's RTX real-time extensions for Windows (including Windows XP and XP Embedded).

To support VxWorks, DDC-I has mapped the SCORE real-time kernel to VxWorks, mapping all SCORE system calls to VxWorks calls, and mapping Ada multitasking to VxWorks multitasking. This enables mixed Ada, C, and Embedded C++ applications developed using SCORE to be hosted under VxWorks on the target system. SCORE is also integrated with Wind River's Eclipse-based Workbench, which gives SCORE developers targeting VxWorks seamless access to the Workbench toolkit.

RTX provides deterministic real-time multitasking, interrupt handling, and other real-time features for Windows applications. To support Windows/RTX run-time environments, DDC-I has mapped Ada tasks to RTX threads. This enables mixed Ada, C, and Embedded C++ applications developed using SCORE to run in real time on Windows systems. In this scenario, RTX provides real-time services that enable designated time-critical Windows applications to process sustained interrupt rates of up to 30 kHz. SCORE version 2.7 provides full support for VectorCAST 4.0, the latest version of Vector Software's software test tool. VectorCAST automates the test process for C/C++, Embedded C++, and Ada 83/Ada 95 program modules, significantly reducing the time, effort and cost required to validate safety-, mission-, and business-critical systems. Version 4.0 adds support for integration testing, which enables designers to test entire subsystems and applications using the same tools available for unit testing. SCORE version 2.7 is available immediately. 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. For more information regarding DDC-I products, contact DDC-I at 1825 E. Northern Ave., Suite #125, Phoenix, Arizona 85020; phone (602) 275-7172; fax (602) 252-6054; e-mail sales@ddci.com or visit www.ddci.com

Headway Software — Headway reView

Ada Reverse Engineering and Static Analysis with Headway Review

Date: July 7, 2006

Headway reView uses advanced reverse engineering and static analysis techniques to create one of the most powerful code comprehension, code review, and source code visualization tools for Architects and Team Leads. It lets you understand the complexity of your software. It gives your business agility by ensuring flexibility in your software assets. Ensuring an optimized software design, it illuminates the period of "radio silence" that accompanies implementation by enabling

senior development staff to analyze the software directly and thus giving technical leads and project managers more control.

Key features include source code visualization, advanced code browsing, source code metrics, dependency management, complexity analysis, snapshots, architectural differencing, change analysis and predictive refactoring (also referred to as pre-factoring).

Headway reView delivers practical, useful visualization of a code-base that are guaranteed to be totally up-to-date and accurate. Analysis and research tools allow developers to thoroughly engage with their source code and design, so that their evolution is constantly understood and controlled. Intelligent extraction (analysis) and intelligent presentation (visualization) combine to give you Software Intelligence.

Headway reView parses Java, J2EE, and Ada to reverse engineer a visual representation of the composition and dependencies of an application. The Headway reView diagrams, called Higraphs, are intuitive, interactive, visual environments in which a developer or designer can gain a truer understanding of how their applications are structured, or more significantly, how their applications SHOULD be structured.

Being code-centric, Headway review has access to all the information about a code-base. There is no separation of code-base and design. The code IS the design. The results of code-base analyses are presented in the model so that the developer can not only find problems, but also understand the context in which they occur.

"Complexity kills. It sucks the life out of developers, it makes products difficult to plan, build and test. … Each of us should … explore and embrace techniques to reduce complexity." Ray Ozzie, Chief Technology Officer, Microsoft Corporation.

Important Note: Headway Software is currently in the process of upgrading Headway Review to our new product line, Structure101. All new Review customers will be supported throughout the migration process and will be provided with complimentary upgrades to Structure101 as soon as it is available for your required programming language.

We plan to complete the migration to Structure101 with the Java version first, before implementing backends for Ada, C/C++ and .Net. For Java we recommend at this point you start with Structure101.

For Ada, we recommend Headway Review.

Praxis HIS — zero-defect security software

Praxis High Integrity Systems produce zero-defect security software for US National Security Agency

12 June 2006

Recent security work carried out by Praxis has now been cleared for general publication by the US National Security Agency (NSA).

The NSA commissioned Praxis to develop secure software for an experimental biometric access control system to meet or exceed Evaluation Assurance Level (EAL) 5 (out of 7) in the Common Criteria. The Common Criteria is an international security scheme aimed at providing confidence to users of security products. EALs 5-7 represent the highest levels of security assurance.

The NSA commissioned this work to evaluate, under controlled conditions, the suitability of Praxis's Correctness by Construction (CbyC) software development method for the development of high-security systems. Praxis and its clients have used CbyC for fifteen years to develop high-integrity software, and the NSA wanted to carry out its own evaluation.

The software developed by Praxis was tested independently of both Praxis and the NSA. During independent test and subsequent use, zero defects were reported. Development costs were lower than traditional methods per line of code.

Keith Williams, Praxis Managing Director, commented “I'm delighted that we are now able to publish the results of this work, which provide further evidence for the cost-effectiveness of Praxis's software development method for high-security software”.

The work is reported in the paper “Engineering the Tokeneer Enclave Protection Software”, co-authored by Praxis and the NSA, and published in the Proceedings of the IEEE International Symposium on Secure Software Engineering, held in March 2006 in Arlington, Virginia, USA. This paper is available from the publications section of the Praxis website.

McKae Technologies — DTraq

From: Marc A. Criley <marccriley@earthlink.net>
Subject: Announce: DTraq Released
Date: Thu, 15 Jun 2006 14:20:17 GMT
Newsgroups: comp.lang.ada

McKae Technologies announces the release of DTraq versions 1.000 for GNAT 3.15p and GNAT 1.100 for GNAT GPL 2005. DTraq is available on the redesigned McKae Technologies website at http://www.mckae.com/dtraq.html.

DTraq is a data logging and playback debugging tool providing near realtime data logging and analysis to aid debugging and validation. Captured, or 'tapped' data from a program can be viewed live while the program is running or, since it is being logged to a file, played back or printed out later for off-line review and analysis.

DTraq differs from other logging and playback tools in that no data layout maps or byte interpretations or "data dumpers" need to be manually created. Nor is the application responsible for converting the raw binary data to text form before logging it. DTraq handles all conversion automatically by scanning the application's source code, identifying tapped data items, and extracting the information it needs to properly convert and display the logged items-simple scalar items as well as arrays and records. When the layout of data items change, rescanning automatically picks up the changes.

Here are the most significant changes to DTraq since the last public release, 0.986a:

- No more code generation and associated compilation needed when building a logging server. The logging server now uses an XML formatted configuration file to recognize and process tapped data items.

- The command line driven "mkdtq" has been replaced with dtq-analyze, providing a GUI-driven interface for source code scanning and generation of the aforementioned configuration file.

- All the DTraq applications are now prefixed with "dtq": dtq-analyze, dtq-vdt, and dtq-dv.

- Data transfer between tapped clients and the logging server has been changed from a stream-based model to one of simply sending tapped data items' bytes through a socket. The result has been a significant throughput increase, and, in conjunction with the use of configuration files instead of code generation, a much simpler implementation.

- The data viewer can now be started independently of the logging server, and the location of the logging server can then be interactively specified. In addition, the data viewer can switch from one logging server to another.

- The need for temporary working directories has been reduced, and the remaining use has been streamlined to reduce the chance of inconsistency errors.

DTraq 1.000 for GNAT 3.15p is licensed using the GNAT-Modified GPL, i.e., GMGPL.

DTraq 1.100 for GNAT GPL 2005 is licensed using the full GPL, although the instrumentation portion that is compiled

into the client remains GMGPL, for what it's worth.

A release for GNAT GPL 2006 will be provided once its downloading server's traffic load eases.

[See also same topic in AUJ 26-2 (Jun 2005), p.80. —su]

From: Marc A. Criley <mc@mckae.com>
Subject: Re: Announce: DTraq Released
Date: Sat, 17 Jun 2006 13:30:05 GMT
Newsgroups: comp.lang.ada

> Is that a change in policy or do you think/know that Code released for use with GNAT 2006 cannot have the linking exceptions (for this part of the code at least)?

The licensing of DTraq is being driven by the licensing of the GNAT Run-time library.

I made an effort to keep the client instrumentation packages GMGPL (by replacing GNAT.Sockets in 3.15p with AdaSockets) so that one could at least theoretically use DTraq on non-GPL code. Frankly I doubt the practicality of that, though, since DTraq uses ASIS, which is compiler-version specific, so to use the GNAT GPL 2005 version of DTraq one would have to compile and build their code with that compiler…which would require the code be GPL if it was to be distributed.

I haven't acquired GNAT GPL 2006 yet, but I have every expectation that it, too, will be GPL.

Continued DTraq development will be for the GNAT GPL 200x compiler versions, since those are AdaCore blessed and maintained. Unless of course someone contracts for a GNAT Pro or other version ☺

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Announce: DTraq Released
Date: Sat, 17 Jun 2006 16:30:00 +0200
Newsgroups: comp.lang.ada

> Continued DTraq development will be for the GNAT GPL 200x compiler versions, since those are AdaCore blessed and maintained. Unless of course someone contracts for a GNAT Pro or other version :-)

I have downloaded the sources (not the binaries) of GNAT GPL 2006 Edition and I confirm that ASIS is pure GPL.

In addition, I am preparing a Debian package of ASIS 2006 under pure GPL. This is in contrast to the GNAT run-time library and every other library, which will remain GMGPL.

So, if someone makes a Debian package of DTraq, that will be pure GPL.

From: Marc A. Criley <mc@mckae.com>
Subject: Re: Announce: DTraq Released
Date: Sun, 18 Jun 2006 20:29:28 GMT
Newsgroups: comp.lang.ada

> For my own OSS work, my position is:
> * code meant for a user to include in her product, GMGPL
> * code intended for tutorial/example/prototype, no restriction
> * code that's part of the toolset, GPL

Though I've not explicitly codified it before as a position, these bullets are consistent with my approach, though I might perhaps be a little more explicit by adding:

* code _available_ for a user to include in her product(s), GMGPL

These covers the situation for XPath In Ada (XIA) and XML EZ Out, which were written because of DTraq, but since they have areas of application beyond that product were broken out on their own for community use.

> As a potential customer of yours, I would want the part of your code that's linked with mine in my product (your runtime) to be GMGPL, regardless of the compiler you use to develop it[1].

Which it now is. I removed the GNAT dependencies from that code, which was primarily the replacement of GNAT.Sockets with AdaSockets, so as to get GMGPL instead of GPL (for GNAT GPL 2005 and beyond).

> Unless I've misunderstood DTraq, though, the recorder side _isn't_ in the same boat, it's not intended to be released to my customers; so the licence terms aren't so crucial. Clearly the binary distribution terms have to match the compiler runtime, but why should the source code?

That is true, the source code license terms aren't so crucial, the code could turn out to be tailored for a specific compiler and runtime, but distributed solely as source code, and therefore merely "inherit" the licensing of the RTL with which it is linked. I opted for GPL on the DTraq core components because it gave me the most flexibility: I could use AdaCore's free software releases, GNAT GPL 2005 and now 2006; I can utilitize any GPL or GMGPL or LGPL licensed software that I find useful; and I don't have to worry about tracking what software is under what license and who might "own" portions if modifications or enhancements were done under contract.

> That said, I can't see any reason why the 'instrumenter' and 'recorder' parts of DTraq shouldn't be pure GPL anyway, since they're not intended to be part of my distribution; and even if they were I could comply with GPL terms for them without affecting my own product. I guess it might be different if my work and the recorder were integral parts of my overall product, but that doesn't seem very likely.

The 'recorder', i.e., the DTraq Logging Server is pure GPL. The 'instrumenter' portions, DTraq.Tap, et.al., could be GPL, but their removal would be required before the distribution of your product if you're licensing with something less than GPL. However, I've suggested that the instrumentation be retained in the product because the taps can be disabled and your product run without a logging server present, and that way you retain the ability to run a deployed application in a remote debug or monitoring mode. And also, if your app requires integrated logging, DTraq can fill the bill for that as well.

> [1] Do you think there's any issue with generated code? One might think that fragments of text copied into generated code could carry licence implications with them.

From the DTraq perspective this is not an issue because the only part that would be at all likely to show up in generated code would be _instantiations_ of DTraq.Tap, and the GMGPL already covers that.

Ada and GNU/Linux

Debian Etch Trasition

Author: Ludovic Brenta
Subject: Ada dans Debian : transition vers GCC 4.1 pour Etch
Date: July 12, 2006
URL: http://www.ada-france.org/article124.html

[Translated from French –su]

Debian is, among other things, an excellent platform for the development and the execution of Ada programs. In the current version, named “Sarge”, that platform uses GNAT 3.15p. In the next one, called “Etch”, the compiler with be GCC 4.1.

Debian 3.1 Sarge is the current and stable version for Debian and it is the one recommended for production. The version currently under test, Etch, will become stable as of December 2006.

At the beginning of July 2006 the Ada component of it however has entered a “zone of turbulence”, hence that version at present is no longer recommended for production. The current turbulences are due to the changes in the compiler, which I explain below.

The decision to use GCC 4.1 rather than the latest version GNAT GPL 2006 Edition results from two concerns:

- the licence of the runtime library should permit redistribution of programs compiled under licences other than GPL. That it possible with GCC but not with GNAT GPL Edition

- interoperability with other languages should be optimal. With GNAT GPL Edition we only have C and Ada. With

GCC with also have C++, Fortran 95, Java, Objective C, and Objective C++.

GCC 4.1 incorporates most part of the new features of Ada 2005, but not all; complete coverage will be attained in the next version.

In Sarge, GNAT supported i386, Sparc and PowerPC. Etch will add Amd64, Hppa, Ia64, Kfreebsd-i386, Mips, Mipsel and s390. There will not be however support for dual architectures such as i386-Amd64 or PowerPC-PPC64. Again, support for that will be incorporated in the next version of Debian, in conjunction with the general support for multiple architectures (multilib).

The gnat-4.1 package has made into Debian Etch in April 2006. After a stabilization period, in July it will become the default compiler.

The new gcc-defaults package, which provides "gnat", is already present in the current unstable distribution in place of the previous "gnat" that was at version 3.15p.

The "gnat-4.1" package is constructed from the source "gcc-4.1" package much like all other languages except for Java. In this way I have joined the team of those responsivble for GCC in Debian.

The other versions of GNAT, including 3.15p and those includes in GCC 3.3, GCC 3.4 and GCC 4.0, will be retired from Debian: I will concentrate myself on a single version so as to be able to provide quality support.

All the other packages presently are in the migration phase. As the binary interface changes all the libraries have to be recompiled with a change in their so-name. Considering the difficulties we are facing the migration should still take another couple of months.

- ASIS: version 2005 GPL Edition has arrived in the unstable version on 2006-07-11 and it should shortly migrate towards Etch. It works well with GCC 4.1, in contrast with 2006 GPL Edition, which requires changes in the compiler.

Pay attention to the changes in the licence; the GNAT-Modified GPL becomes pure GPL. The documentation, which was non-libre (in that it does not allow modifications) has been transferred to the new package asis-doc.

- GtkAda: v2.8.1 will shortly be integrated with the unstable version, together with a change to the GPL pure licence.

- GPS is transitioning from version 2.1 to version 4.0 using GtkAda 2.8.1.

- A GDB improved for Ada is migrating from version 5.3 to version 6.3, including support for the new format utilised by GCC for debug information (DWARF2 in place of STABS).

- Florist, the POSIX interface, is going towards the 2006 GPL Edition with a change to the pure GPL licence.

- GLADE (the support for distributed systems) is transitioning from 3.15p to the 2006 GPL Edition with change to the GPL pure licence.

- XML/Ada is moving from 1.0 to 2.2 with change of licence to pure GPL.

- AWS is moving from 2.0p to 2.2 with change of licence to pure GPL.

- Charles will be retired from Debian: that library of generic containers is replaced by the normalized containers (Ada.Containers) supported by GCC 4.1.

- GNADE, the interface to ODBC, PostgreSQL, MySQL and SQLite databases will go from v1.5.1 to the latest to date, 1.6.1. It will stay with GMGLP.

- AUnit will go from version 1.01 to 1.05 without changes of licence as that has always been pure GPS.

- AdaBrowse has already been recompiled with ASIS 2005 and it is already part of the unstable version.

- The destiny of the other libraries, more rarely utilized, remains to be decided. The group includes libadabindx, libtexttools and libopentoken. Please contact me in case you needed to use them and wished they should stay in Debian.

Change of licence

The change of licence towards pure GPS, which affects most important libraries, has been made necessary by their provider, AdaCore. With pure GPS it is no longer possible to distribute non-libre software that uses those libraries. If your software uses a library under pure GPL, you have four possibilities:

- to not distribute your software at all (you may of course use it privately)

- to distribute your software under GPL, hence with open source and the four liberties (inspection, redistribution, modification and retristribution of modified copies)

- to distribute only your open source under a licence of your choice (for example with clauses for confidentiality and non-redistribution) and to ask your users to recompile it themselves

- contact AdaCore and buy a GMGPL licence that allows you to distribute your software without devolving your sources.

It is possible to safeguard the old versions under GNAT-modified GPL along with the new ones. Howevr that requires some extra maintenance effort, which I cannot and do not want to provide all by myself. I am ready to provide two versions (GPL and GMGPL) of some specific libraries in parallel if and only if I receive some financial support. This is thus a call for financial contributions: contact me in case you needed GMGPL libraries. That goes especially for GtkAda, AWS and XML/Ada.

References:

- Debian : http://www.debian.org

- Debian Ada Policy : http://www.ada-france.org/debian/debian-ada-policy.html

Ludovic Brenta is the principal responsible for the Ada packages in Debian. He works as a volunteer since 2003 and provides a complete and integrated platform for development and execution. You can contact him at: lbrenta@debian.org.

Multiple Debian Environments

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Is there a GMGPL GtkAda available for use with GNAT 3.15p?
Date: 18 Aug 2006 03:37:22 -0700
Newsgroups: comp.lang.ada

> Can anyone tell me if there is a GtkAda under the modified GPL that I can use with 3.15p, and if so, where I can obtain it?

From Debian or gnuada.sourceforge.net, of course. I think the gnuada packages are designed for parallel installation on several platforms.

> I had an old source archive and I've spent all evening trying to build it. I've all the prereqs (possibly at levels that are too high) but I don't have things installed in standard paths (Linux) because I'm running multiple copies of GCC on this machine. I created a user just for working with the 3.15p toolset. I can build vanilla Ada sources and ncurses applications without difficulty so the base 3.15p and ncurses installations are fine. What I'm unable to do is to build GtkAda in this environment.
> I built glib, gettext, and Gtk+2.6 (several times each) and I still get all kinds of errors when I try to build GtkAda 2.4.0 because it says I don't have some of these things installed. When I built almost every .adb in Emacs using gnatmake from Ada-mode everything compiled properly but then I couldn't bind any sample programs because of unresolved symbols (missing libraries).
> If any kind soul would help with this that would be grand.

Since you're obviously willing to spend as much time as necessary, why don't you go all the way and solve your problem once and for all? If not already done, install Debian Sarge (with GNAT 3.15p and GtkAda 2.4 under GMGPL in the standard paths), then create a chroot environment containing Etch or Sid (with GCC 4.1 as the Ada compiler, but no GtkAda yet). You can create as many chroots as you like. Use debootstrap to

create each of them, then use apt-get inside each chroot to install packages as usual.

Then, bind-mount your /home directory in each of the chroots, so you share it between all chroots and the top-level root.

Then, copy your top-level /etc/passwd into each of the chroots, so all users of the top-level root also exist in the chroots.

Last, use dchroot to allow non-root users to use chroot in a controlled way.

This solution is less error-prone, quicker, and much much more maintainable than recompiling everything. I use it myself on my machine: Etch with gnat 3.15p in the top-level root, Sid (unstable) with GCC 4.1 in a chroot where I do the Ada transition.

Linux Kernel Modules

From: Frank <franjoe@frisurf.no>
Subject: call procedure in Linux-Ada-module from "normal" program
Date: Thu, 24 Aug 2006 18:33:21 +0200
Newsgroups: comp.lang.ada

Debian sarge kernel 2.4.27, gcc-3.3

I've been tinkering with modules by using Ada. The module compiles, and I manage to do insmod on it. The symbols appears in ksyms; like "some_call_me".

Now I wish to call "some_call_me" from the test program (3) — is that possible?

From: Manuel Gomez <mgrojo@gmail.com>
Subject: Re: call procedure in Linux-Ada-module from "normal" program
Date: 25 Aug 2006 04:10:13 -0700
Newsgroups: comp.lang.ada

Maybe this text is useful for you, but don't know if you already follow those steps.

Section "Writing Linux Modules" from the "The Big Online Book of Linux Ada Programming":
http://www.pegasoft.ca/resources/boblap/16.html#16.16

Generating Ada from UML

From: "Martin Krischik" <krischik@users.sourceforge.net>
Date: Mon, 07 Aug 2006 14:40:45 +0300
Subject: Re: Generating Ada from UML on Linux
Newsgroups: comp.lang.ada

> […] Are there any Linux tools which can generate Ada code from UML diagrams? Maybe even some Open Source tools?
> I have found a package called "dia2code" on my Ubuntu system. Does any of you have any experience with using it to generate Ada code from UML diagrams? I have made some very simple experiments with it, and it appears that it cannot express private attributes and operations properly in Ada. Is that just me making a mistake? Or is it an error in `dia2code`?

Well we also have xmi2code [1] which works with Umbrello [2] as UML tool.

[1] http://xmi2code.sourceforge.net/

[2] http://uml.sf.net/

From: "okellogg" <okellogg@freenet.de>
Subject: Re: Generating Ada from UML on Linux
Date: 7 Aug 2006 06:08:12 -0700
Newsgroups: comp.lang.ada

While dia2code and xmi2code are close to unmaintained these days, Umbrello is alive and kicking. With Umbrello you stand a much better chance of getting possible problems fixed.

From: Jacob Sparre Andersen <sparre@nbi.dk>
Subject: Re: Generating Ada from UML on Linux
Date: Mon, 07 Aug 2006 15:18:58 +0200
Newsgroups: comp.lang.ada

It appears that the Ada code generator is included in the Ubuntu package "Umbrello". I would also say that it seems to work much more like I would expect an UML to Ada converter to do.

Linux GUI Bindings

From: Simon Clubley <clubley@eisner.decus.org>
Subject: Ada bindings to Linux GUI toolkits?
Date: 30 Jun 2006 19:11:17 -0500
Newsgroups: comp.lang.ada

With the change in license for GtkAda, I have started looking for Ada bindings for other GUI toolkits that run under Linux.

At:

http://www.adapower.com/index.php?Command=Class&ClassID=AdaGUI&Title=Ada+GUI

there is a list of known bindings and only the base X11 and Motif interfaces appear to have Ada bindings for Linux. (GWindows appears to be MS-Windows only).

Are any other Ada open source bindings to current Linux based GUI toolkits available?

From: Simon Wright <simon@pushface.org>
Subject: Re: Ada bindings to Linux GUI toolkits ?
Date: Sat, 01 Jul 2006 10:10:59 +0100
Newsgroups: comp.lang.ada

http://tcladashell.sourceforge.net/ (no downloadable distribution as yet, checkout via anoncvs or follow the links to Terry Westley's original site).

From: Lucretia <lucretia9@lycos.co.uk>
Subject: Re: Ada bindings to Linux GUI toolkits?
Date: 1 Jul 2006 08:25:43 -0700
Newsgroups: comp.lang.ada

Well, I have some source that I'm in the process (slowly) of making available. It's not complete but does work. There needs to be a lot of work done on it, especially on the C++ virtual functions.

You can keep an eye on it here:

http://wxada.tigris.org

http://wxada.tigris.org/screenshots.html

From: Simon Clubley <clubley@eisner.decus.org>
Subject: Re: Ada bindings to Linux GUI toolkits?
Date: 3 Jul 2006 06:36:05 -0500
Newsgroups: comp.lang.ada

> Also learning Gtk (any GUI Toolkit) well is hard. One probably would like to preserve one's investment in that area, so moving to another language (and staying with Gtk) is a definite option. My recommendation at that point for lovers of exotic languages would be Ocaml :-), but YMMV.

I was thinking about this over the weekend and have now come to the same conclusion, but I would like to thank people for their pointers to Ada bindings to other toolkits anyway.

My current GtkAda programs tend to be things like microcontroller programmers/debuggers or status monitors for things (like a current autonomous robot experiment) that do need Ada. Neither of those GUI applications really require Ada (the robot has to handle the status monitor link potentially failing for reasons unrelated to the code in the status monitor anyway), and I've spent quite a bit of time learning the GTK toolkit.

You are correct about this affecting hobbyist users. The above projects (and others) are hobbyist projects, but since I don't know what I will do with this (and other) projects in the future, I would like to be in control of how I use my code.

I think that the most annoying thing is that GtkAda started out life as GMGPL[1] before been changed to pure GPL and then impacting all the projects built up until then. If it had been GPL at the start, then that would have been known _before_ designing and writing any code.

[1] It most definitely started out life as GMGPL. From the GtkAda mailing lists:

> http://lists.adacore.com/pipermail/gtkada/2002-February/001204.html
>
> Arnaud Charlet
>
> Sun, 10 Feb 2002 00:38:36 +0100
>
> GtkAda is licensed under the GNAT modified GPL (see the GtkAda spec files more the exact wording). So basically you should be fine, but that being said, whether you can use it in such or such exact conditions would require to examine all the licenses involved or to hire a lawyer to do this job for you.

From: M E Leypold <kontakt@m-e-leypo
Subject: Re: Ada bindings to Linux GUI toolkits ?
ld.de>
Date: 04 Jul 2006 02:04:55 +0200
Newsgroups: comp.lang.ada

The GMGPL interested community would have to maintain (i.e. bugfix and conservatively extend) the GMGPL version for the next years. Is it big enough?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Ada bindings to Linux GUI toolkits?
Date: 4 Jul 2006 03:19:08 -0700
Newsgroups: comp.lang.ada

The number 1 requirement, as far as Debian is concerned, is that GtkAda must be suitable for supporting GPS.

GtkAda >= 2.4.0 is fine with GPS 2.1 (this is in Sarge).

GtkAda >= 2.8.1 seems to be fine with GPS 4.0.0 (this is the plan for Etch).

Notably, GtkAda 2.4.0 would NOT be enough for GPS 4.0.0.

Just a data point.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Ada bindings to Linux GUI toolkits?
Date: Wed, 05 Jul 2006 21:37:21 +0200
Newsgroups: comp.lang.ada

> Which of course doesn't require GMGPL. (I expect this has been said already, but ..)

No, but it implies that it may be impractical to support a GMGPL version (namely 2.4.0) of GtkAda in Debian.

From: M E Leypold <kontakt@m-e-leypold.de>
Date: 05 Jul 2006 00:55:07 +0200
Subject: Re: Ada bindings to Linux GUI toolkits?
Newsgroups: comp.lang.ada

> Or else live with the last GMGPL version.

"Live with" without the option of conservative maintenance sounds pretty much like a dead end. One reason to decide for the main stream (Gtk) is the continued development, in backend and interface. Writing now a software based on something that will become more and more unbuildable within the next years seems to be a waste of time.

Personally I never ever build infrastructure on something that is not maintained any more or doesn't have community debugging it. Considering all the answers that I got on the bugs in Gnat 3.15p (which is after all in current Debian) that told me, that 3.15p is dead dead dead, I can imagine what answers anyone will get (here in c.l.a) when asking about GtkAda 2.4.0 problems in some months: It will go like this:

- "Got problem with GtkAda 2.4.0 […] "

- "Oh man / good grief, that has been fixed looong ago. Just use a current GtkAda."

- "There is no newer GMGPL GtkAda"

- "The new version is GPL, just take that?"

- "Can't use GPL"

- "Why not?"

- "I, want to link closed source."

- "Just buy support or free your software! You wouldn't want to restrict the freedom of your customers?"

- "But …"

From there the discussion quickly spins out of control, without actually addressing the initial technical problem again.

Mind you, that I'm not reproaching anyone that the recent discussions went like they went. They were rather instructive to me. But the thought experiment "What will I do when I think I found a bug in an already dusty version?" should make clear, that using an old version will not really be fun.

From: Jeffrey R. Carter <spam.not.jrcarter@acm.not.spam.org>
Subject: Re: Ada bindings to Linux GUI toolkits?
Date: Thu, 06 Jul 2006 03:43:11 GMT
Newsgroups: comp.lang.ada

A fairly common practice on real projects is to choose a toolset and freeze it for the life of the project. Such projects "live with" whatever drawbacks their chosen toolset may have. So, living with an older toolset is not that uncommon a situation.

Official Debian Developers

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Ada in Debian: I am now an official Debian Developer
Date: 4 Jul 2006 05:03:52 -0700
Newsgroups: comp.lang.ada

Subject says it all. […]

The full process took 35 months since I started packaging gnat in Debian. I'm sure glad it is complete at last. Also, this will give me read-write access to the Subversion repository that holds the GCC build scripts. I'm not going to take advantage of that privilege just now, as my work on GCC 4.1 is complete. On the other hand, I anticipate I'll make quite a few package uploads in the next few weeks, without the need for a sponsor. Stay tuned for more announcements :-)

Ah, one last thing: I am now in a position to sponsor other people who would like to contribute to Debian. So, if you too would like to maintain a few packages (or co-maintain them with me), please drop me a note.

From: george <gshapovalov@gmail.com>
Subject: Re: Ada in Debian: I am now an official Debian Developer
Date: 4 Jul 2006 12:28:44 -0700
Newsgroups: comp.lang.ada

Brothers in arms, rejoice!

Wow!

And I thought we (Gentoo) were slacking, with mentoring/recruitment process taking 3-6 month nowadays :). Well, OK, most of our recruits have a history (usually upwards of a year) of submissions to our bugtracking system, but still.. Apparently you cannot beat Debian when it comes to, um, I don't know, — the process? :)

Congratulations!

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Ada in Debian: I am now an official Debian Developer
Date: Tue, 04 Jul 2006 21:38:54 +0200
Newsgroups: comp.lang.ada

Well yes. Gentoo is growing faster than Debian, and I think that's one reason. No one is paid to process applications, and there are many, and the process is indeed very thorough. Just to give you an idea, the very first step is to have your GPG key signed by two Debian Developers or more.

From: Poul-Erik Andreasen <poulerik@pea.dk>
Subject: Re: Ada in Debian: I am now an official Debian Developer
Date: Mon, 17 Jul 2006 03:58:56 +0200
Newsgroups: comp.lang.ada

Congratulations for me too.

I want you to know that it actually was your work which made me change my preferred distro to Debian. And a have no regrets at all :-)

GNU/kFreeBSD GNAT port

Title: GNAT porté vers GNU/kFreeBSD
Date: mercredi 31 mai 2006.
Source: Association Ada-France
URL: http://www.ada-france.org/breve50.html

Aurélien Jarno has ported GNAT 3.4, 4.0 and 4.1 to GNU/kFreeBSB. The patch has been accepted in GCC. He has also ported GNAT 3.15p to that system; in Debian the gnat 3.15p-18 package incorporates the necessary modifications. The other Ada packages currently in Debian will continue to exist.

GNU/kFreeBSB is a platform built on the FreeBSB kernel but with the GNU user environment (glibc, utilities, etc.).

A Debian GNU/kFreeBSB distribution currently is under development.

The Debian gnat-3.4, gnat-4.0 and gnat-4.1 are lready included in that.

GNAT in FreeBSD

From: M E Leypold <kontakt@m-e-leypold.de>
Date: 02 Jun 2006 17:57:59 +0200
Subject: GNAT at FreeBSD -- Question about available versions …
Newsgroups: comp.lang.ada

> I'm trying to port GCC 3.4.6 with Ada support to FreeBSD.

BTW: I remember that 3.15p has been in earlier FreeBSD-Release. But when I looked into Release 6.x I only found gnat-2005 (ACT GPL GNAT, I think).

Does anybody here know, wether there is any technical reason that Gnat-3.xp has been dropped from the 6.x release stream (i.e. problems between the tasking runtime and FreeBSD thread model or whatever) , or wether the only reason is that it was just too much effort or no maintainer available?

From: Karel Miklav <karel@lovetemple.adbloccker.net>
Subject: Re: GNAT at FreeBSD -- Question about available versions …
Date: Fri, 02 Jun 2006 19:13:42 +0200
Newsgroups: comp.lang.ada

GNAT 3.15p wasn't dropped, it was just updated to GNAT 2005. Check the revision history on i.e.: http://www.freshports.org/lang/gnat. There is still a binary version of the old compiler on servers used to bootstrap the new one.

There are no direct technical problems just a lack of manpower. Maybe you can help — try to download and play with my port than share your experience (beware, it doesn't deinstall).

GNAT for Mac OS Tiger

From: Burkhard <burkhard@no.spam.please>
Subject: Re: gnat for MacOS Tiger?
Date: Sun, 13 Aug 2006 21:56:03 +0200
Newsgroups: comp.lang.ada

> I just discovered that the GCC that comes with Tiger does not have Ada support (unlike the one that comes with SUSE Linux). […]
> So what does one need to do these days to get an Ada-capable GCC on OS X 10.4?

Take a look at www.macada.org

From: Simon Williams <williams@ntlworld.com>
Subject: Re: gnat for MacOS Tiger?
Date: Mon, 14 Aug 2006 02:20:01 GMT
Newsgroups: comp.lang.ada

[…] Mac Ada is alive and well and we have GCC 4.2 from the FSF archives for tiger that works quite well as well as a plug-in for apples IDE (Xcode).

On Intel macs Paralleles makes a really nice virtual machine capability and I run Mac OS, Windows XP, and Linux (Ubuntu and Fedora) all on the same machine. with enough memory they will all run at once.

I did some GtkAda code and built it and had same app running under mac and windows at the same time. Worked very nicely to test it out.

From: Javier Miranda <jmiranda@iuma.ulpgc.es>
Subject: Re: gnat for MacOS Tiger?
Date: Mon, 14 Aug 2006 04:39:21 -0400
Newsgroups: comp.lang.ada

Go to the Libre site of Adacore (https://libre2.adacore.com/), select and download the ppc-darwin platform. Remember that if you want to execute GPS you also need to install the X11 server, available at

http://www.apple.com/downloads/macosx/apple/x11formacosx.html

…. and launch GPS from an X11 console.

From: Simon Williams <williams@ntlworld.com>
Subject: Re: gnat for MacOS Tiger?
Date: Tue, 15 Aug 2006 01:57:30 GMT
Newsgroups: comp.lang.ada

But keep in mind that if you get the version from AdaCore you can only use it for producing Open source applications. the 4.2 compiler at macada.org is built from the FSF sources so it retains the GMGPL exception.

Besides ours works with Apple's IDE Xcode. ;-)

I have just put out beta installers. Join the mailing list if you want to get them. They will be on the website in a week or so after we are sure they are OK.

References to Publications

Dedicated STSC CrossTalk

From: roderick.chapman@googlemail.com
Subject: Ada2005 in August's CrossTalk
Date: 12 Jul 2006 09:00:45 -0700
Newsgroups: comp.lang.ada

I just noticed this:

http://www.stsc.hill.af.mil/crosstalk/theme.html

It seems Ada 2005 is the main theme for the August 2006 issue of CrossTalk.

Anyone know who's contributed to this issue?

From: Britt Snodgrass <britt.snodgrass@gmail.com>
Subject: Re: Ada2005 in August's CrossTalk
Date: 25 Jul 2006 12:51:17 -0700
Newsgroups: comp.lang.ada

This issue is now on-line at

http://www.stsc.hill.af.mil/crosstalk/2006/08/index.html

It looks like a good read.

NSA's Tokeneer system

From: roderick.chapman@googlemail.com
Subject: ANN: NSA, SPARK, Praxis project — results now available
Date: 12 Jun 2006 08:32:54 -0700
Newsgroups: comp.lang.ada

I'm pleased to say that we're finally able to publish the results of our work on the NSA's Tokeneer system. Press release and PDF of the full paper from the recent ISSSE Conference are at www.sparkada.com as usual.

— Rod, SPARK Team

JGNAT and MGNAT

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: Re: C to JVM, time to revive JGNAT?
Date: Wed, 09 Aug 2006 11:48:23 +0200
Organization: Adalog
Newsgroups: comp.lang.ada

> AMPC (Axiomatic Multi-Platform C): http://www.axiomsol.com/

If you are interested in non-Java languages for the JVM, have a look at http://www.robert-tolksdorf.de/vmlanguages.html

That's really impressive…

From: Colin Paul Gloster <Colin_Paul_Gloster@acm.org>
Subject: Re: C to JVM, time to revive JGNAT?
Date: Wed, 9 Aug 2006 12:59:09 +0200
Newsgroups: comp.lang.ada

From the homepage of Axiomatic Solutions:

"If you write Java code, you can write once and run anywhere!"

and

"I write C code, and with AMPC I can write once and run anywhere!"

and

"The best and easiest way to convert C programs to standard Java bytecodes (classes) is AMPC from Axiomatic Solutions"

"AMPC allows users to develop software using the standard C programming language.

AMPC covers a very large subset of ANSI C (1989). A notable difference is that "double" in AMPC is 32 bits long. In order to utilize 64-bit floating point you can use "DOUBLE".

From: napi@axiomsol.com
Subject: Re: C to JVM, time to revive JGNAT?
Date: 9 Aug 2006 19:53:32 -0700
Newsgroups: comp.lang.ada

The compiler/IDE itself comes in three versions (Linux, Mac OS X, and MS Windows), but the generated code (.class files) should be able to run on any

properly installed JVM/JRE. In other words, AMPC is a cross-compiler suite whose development platforms are Linux, Mac OS X, and MS Windows, all targeting the platform-independent JVM.

The ANSI C standard does not specify the size of scalar variables since there are various CPU architectures out there from 8 bits to 64 bits, etc. So, making "double" to be 32 bits or even 16 bits is still conformant to the ANSI C standard, although abnormal. AMPC actually supports 64 bit floating point by means of the type DOUBLE.

From: Georg Bauhaus <bauhaus@futureapps.de>
Subject: Re: C to JVM, time to revive JGNAT?
Date: Wed, 09 Aug 2006 12:15:20 +0200
Newsgroups: comp.lang.ada

I find the idea more than obvious. Zillions of mobile computers and the like are driven by JVMs. Major web sites, too. Ada, as has been explained and also proven as early as 1996 by Tucker Taft and colleagues, is a language well suited for targeting the JVM.

I'm using an Ada->J-code compiler, and I must say that in spite of some rough edges (it's not full Ada 95, older byte codes just like JGNAT), it's fun, it's quick, and it works. The Java-Ada integration is very smooth, quite unlike your usual binding. Tons of libraries at your finger tips.

Is it that Ada programmers feel bad about virtual machines? Do they feel more powerful when they can pretend to themselves to be controlling "the metal"?

According to Robert Dewar at FOSDEM, GNAT is going the JNI and/or "OO-ABI" path, which I interpret to also mean that Redhat is funding GCC development for compiling the Java language directly to machine code. GCC C++ ways can be integrated with GCC Ada ways.

But how can Java, compiled to processor instructions, be so helpful where the OS is really a specialized JVM?

Sometimes research going into Real Time Java, Java memory management, object lifetime etc, is really reapeating Ada stories, AFAICT from my limited view. What if the two efforts can be readily integrated, taking advantage of Ada features, JVM libraries, portability, etc etc?

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Wed, 09 Aug 2006 19:01:27 +0300
Subject: Re: C to JVM, time to revive JGNAT?
Newsgroups: comp.lang.ada

I think JGNAT, MGNAT and Interfaces.CPP are interesting technologies but somehow undermaintained.

While MGNAT seem to come along nicely both JGNAT and Interfaces.CPP are stalled on a missing code analyzer/generator for the needed thin bindings. It is just too much work to create them by hand.

And a last point: the GCJ (GNU Compiler for Java) is coming along nicely as well Interfaces.Java (interface to GCC) might be an alternative to JGNAT.

Sadly we don't have the man power for any of these projects. Apart from MGNAT which is sponsored by the US Air force.

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Thu, 10 Aug 2006 13:11:02 +0300
Subject: Re: C to JVM, time to revive JGNAT?
Newsgroups: comp.lang.ada

> Well, [GCJ] wouldn't give us JVM target, or?

GCJ is dual target — binary and jvm.

> Having a naked Ada compiler into anything isn't much useful. Ada's run-time library is the thing that makes it useful.

Let's be honest here: the Java runtime lib is a lot more powerful than Ada's.

Ada Inside

Boeing Company Selects SofCheck

http://www.sofcheck.com/news/boeingpressrelease.html

The Boeing Company Selects SofCheck's Error Detection Technology for Quality Assurance and Testing of Advanced Avionics System

SofCheck Inspector™ for Ada finds flaws early in the development process, assuring quality of today's complex, mission critical applications, like those used in transportation, finance, communications and health care.

BURLINGTON, Mass. (June 13, 2006) – SofCheck, Inc., a provider of software analysis and verification technology, today announced that The Boeing Company has selected SofCheck Inspector™ for Ada to perform analysis and quality assurance on an advanced avionics system. SofCheck Inspector for Ada offers sophisticated error detection which is vital to assuring the quality of today's increasingly complex mission critical applications, like those utilized in the transportation, financial services, communications and health care industries.

"Software powers many of the products and services that we rely on every day—from automobiles and medical devices to ATMs and mobile phones," said Tucker Taft, chairman and CTO of SofCheck. "As applications have become more complex and more safety and mission critical, organizations like Boeing recognize that manual error detection and quality assurance cannot adequately test systems to the level required by their customers and government regulations."

SofCheck Inspector is available for Ada and Java. A complement to traditional run-time testing tools, it helps eliminate programming errors by performing advanced static analysis on the compiled program source code. The SofCheck Inspector family utilizes static control-flow, data-flow, and value propagation techniques to identify places where run-time errors could occur. This automated software quality technique provides 100 percent path coverage and enables identification and elimination of flaws very early in the software life cycle—before run-time testing. The system runs unattended, analyzes about 100,000 lines of code per hour, and does not require any application domain knowledge to configure and run.

About The Boeing Company

Boeing is the world's leading aerospace company and the largest manufacturer of commercial jetliners and military aircraft, with capabilities in rotorcraft, electronic and defense systems, missiles, satellites, launch vehicles and advanced information and communication systems. Boeing's reach extends to customers in 145 countries around the world, and is the number one U.S. exporter in terms of sales. (www.boeing.com)

About SofCheck

Founded in 2002, SofCheck develops technology that enables software developers and IT organizations to detect and eliminate bugs that can cause crashes or numeric overflows earlier in the development cycle, improving overall software quality and reducing time-to-market. SofCheck's flagship product, SofCheck Inspector, is a complement to traditional run-time testing tools, employing advanced static error detection technology and push-button convenience to find lurking defects in software. SofCheck is a privately held company whose clients include: Raytheon, Northrop Grumman and United Technologies.

Tomahawk Mission Planning

Boeing Selects Aonix ObjectAda for Tomahawk Cruise Missile Mission Planning Software

Product maturity, vendor responsiveness, and new capabilities cited as key selection criteria

San Diego, CA, August 28, 2006

Aonix®, a provider of complete solutions for safety- and mission-critical applications, announced that Boeing has chosen and licensed ObjectAda for Windows. Boeing plans to use Aonix's ObjectAda for Windows for ongoing software development and for migration tasks on the Tomahawk Mission Planning (TMP) Software Platform. Boeing's interest in Aonix's ObjectAda for Windows hinges on several technical factors, including its full compatibility with Microsoft's .NET platform.

Facing legacy obsolescence and diminishing support for their existing Ada development environment, Boeing's TMP group initiated a full-scale evaluation of available Ada compiler and tool solutions. Their challenge was to find an Ada vendor with compiler technology able to support a very large Ada source code base, meet stringent performance and functionality requirements, and efficiently support a large software development team. In order to port a large code base without requiring a large investment of new engineering resources, Boeing's TMP group needed a multilanguage development environment to accommodate existing C, Fortran, and .NET software assets.

"Aonix rose to all the challenges we laid out," noted Dan Turpin, TMP Systems Engineer. "They accommodated specific requirements critical to our success, such as performing specific debugger and compiler performance improvements that we needed."

Similarly, Ben Ralston, TMP's compiler technical evaluator, stated, "As an engineer, I realize it was no small feat to accomplish technical changes of this magnitude to their compiler, especially in such a short time frame. Aonix met all of our objectives."

The Aonix ObjectAda for Windows brings the improvements of ObjectAda 8.2 to the Windows development platform. In integrating current Windows improvements with the Aonix Ada 95 compiler, Aonix has delivered enhancements to the object code and symbolic debugging information generation and provided full compatibility with the Microsoft Visual Studio .NET 2003 development tools. Recognizing the growing number of large-scale Ada projects, ObjectAda 8.2 for Windows offers dramatic performance improvements for developers linking executable files or initiating debugging sessions for large programs.

ObjectAda for Windows 8.2 includes the comprehensive Ada libraries needed for calling Windows Win32 and the Visual C++ .NET 2003 MFC interfaces from application source code written in Ada. In ObjectAda for Windows, these Ada binding libraries are fully compatible with the Microsoft Visual Studio .NET 2003 tools and libraries.

As part of the ObjectAda 8.2 family, ObjectAda for Windows allows developers to choose between the traditional Aonix IDE for development and the new AonixADT™ Eclipse plug-in. AonixADT incorporates Ada-project awareness, an Ada-language sensitive editor, Ada-language compile and build capabilities, and a complete Ada debugger interface, enabling Ada developers to enjoy state-of-the-art interface capabilities geared to maximize developer ease and efficiency.

Shipping and Availability

ObjectAda for Windows is available immediately for Windows 2000 and XP platforms. For more information about ObjectAda 8.2 for Windows, please visit: www.aonix.com/objectada.html.

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.

"Health-Conscious" Planes

SCORE® Helping Develop "Health-Conscious" Planes

Product: SCORE®

Imagine a jet able to warn air and ground crews when the plane exhibits the earliest symptoms of a serious problem. This long-term goal in the aviation industry is no longer limited to the realm of science fiction, as engineers under the wings of the FAA Aviation Safety Program use DDC-I's development environment — Safety Critical, Object-oriented, Real-time Embedded (SCORE®) to develop the technology to build "health-conscious" planes: the Aircraft Condition Analysis and Management system (ACAMS).

SCORE's multi-language capabilities allow project engineers to develop safety-critical code in Ada and compile the board support package, originally written in C, within a single integrated development environment. The PowerPC target relies on SCORE's JTAG integration to enable work at the bare board level with minimal intervening software layers, reaching down onto the board to debug by directly manipulating the processor.

Unique in the aviation industry, ACAMS offers real-time diagnosis and prognosis within a complex dynamic system, including flight subsystems, landing gear, and structural elements. Already configurable for propulsion systems, ongoing development is expected to incorporate, or simply merge with, ongoing propulsion system health management programs.

ACAMS consists of onboard and ground-based elements. In the air, proprietary models and algorithms analyze data in real time to identify and help manage anomalies. Results are automatically prioritized in accordance with user-specific criteria to assess the impact of fault conditions on future operation. If a critical anomaly appears, data is automatically transmitted to the ground crew. In certain cases, the system can even predict faults in the monitored subsystems before they occur.

On the ground, ACAMS combines collected and analyzed in-flight data with historical information like component maintenance history and reliability data, as well as quick-access recorder and flight operations quality assurance data for commercial air carriers. Such combined analysis will facilitate improved long-term, fleet-wide aircraft dispositioning and improve maintenance scheduling and parts supply management.

Future research and development is expected to include integration of ACAMS and ACARS (Aircraft Communications Addressing and Reporting System), an air-to-ground network enabling aircraft to emulate mobile computer terminals linked to a ground-based command-and-control management system.

While aircraft accidents caused by equipment failure are responsible for an estimated 23% of serious aviation mishaps, many may someday be predicted and prevented using ACAMS technology developed with SCORE®.

EADS Unmanned Aerial Vehicles

Future EADS UAVs with integrated avionic systems by SYSGO
Publication date on this website: Thursday, July 06, 2006
Company: SYSGO AG
Category: Press Releases : RTOS

Summary:

SYSGO´s PikeOS Kernel enables new platform architecture for military avionic systems

Mainz/Germany, June 6, 2006

SYSGO, the European vendor for safety critical COTS software, facilitates a new system architecture for an open and modular platform for FMS/MMS (Flight Management System/Mission Management System) avionic systems which will be used in future Unmanned Aerial Vehicles (UAVs). SYSGO´s real-time operating system PikeOS will be used as an integration platform in order to integrate various existing avionic systems, formerly based on other safety critical commercial operating systems, with a single COTS hardware running PikeOS.

Full Text:

The main project objective is the integration of applications with different levels of criticality on a single COTS hardware. This will be realized by using SYSGO´s microkernel based, COTS real-time operating system, PikeOS. The `New Avionics Structures´ department of EADS Military Air Systems leads the project. The resulting system is meant to be used in future UAVs. The project is financed by the German Ministry of Defense´s Research and Procurement Agency BWB, supporting research projects in the field of safety critical systems.

The major objective of this project is to reduce the overall costs for avionic systems, by integrating several existing systems on just one platform. Additionally, the new platform architecture opens the door to use Linux based open source components for non-critical applications, which is another important step towards significant cost savings. Using the partitioning concept of PikeOS, safety critical real-time applications can co-exist with Linux, guaranteeing that the Linux applications are not able to compromise the execution of the critical tasks.

The critical legacy applications are based on Ada. These applications will run on top of PikeOS using ObjectAda, the Ada runtime environment from AONIX, which is already available for PikeOS. Other legacy components, which previously were based on other COTS operating systems, will be migrated to a PikeOS API, which emulates the system calls of the legacy RTOS. The reference implementation is based on a 3U cPCI Board which integrates IBM´s 750FX PowerPC.

About PikeOS

PikeOS is a real time operating system supporting the principle of software partitioning.

The foundation of PikeOS is a powerful microkernel, a genuine development of SYSGO AG, developed according to the highest safety standards. Therefore, PikeOS is meant to be used in safety and mission critical systems as well as standard embedded systems for which hard real time is mandatory.

Each software partition of PikeOS can run an entire operating system such as Linux, POSIX and ARINC-653 or native application programs. Several of these heterogeneous partitions can co-exist. Fault containment is one of the features of software partitioning, enabling the use of critical and non-critical software on the same hardware platform. PikeOS is the ideal choice for legacy code migration, using embedded Linux in a hard real time environment or for platform consolidation. To support PikeOS customers, SYSGO included CODEO in the PikeOS suites. CODEO is the Eclipse-based development environment from SYSGO, offering additional plug-ins for system configuration, validation and analysis as well as comfortable application development.

About SYSGO AG

SYSGO is specialized in design, implementation and configuration of system software for the embedded market with a special focus on the operating system environment. SYSGO develops device drivers, board support packages and firmware and supports its international customers with products, services and support for Embedded Linux, real time and certification for safety-critical applications. The target markets are automotive, electronics, board vendors and semiconductors, engineering and medical technology as well as aerospace. SYSGO customers include DaimlerChrysler, EADS Airbus, EADS Military Air Systems, Honeywell, IBM, Raytheon, Rheinmetall, Rockwell-Collins, Rohde & Schwarz, or Siemens VDO. SYSGO AG was founded in Mainz, Germany, in 1991 and was reincorporated as a joint stock company in October 2002. Today, the company employs more than 70 people and has six facilities in Germany and Europe.

About EADS Military Air Systems

Military Air Systems (MAS), the centre of competence for all manned and unmanned combat aerial vehicles within EADS, is an integrated part of the EADS Defence & Security Systems Division (DS). With revenues of about € 5.6 billion in 2005 and roughly 23,000 employees across nine nations, DS forms the defence and security pillar within EADS. As a Large Systems Integrator, it offers integrated systems solutions to the new challenges confronting armed forces and global security. It is active in the areas of manned and unmanned integrated combat and mission air systems, including related training services, and in missile systems, battlefield management systems for all branches, global security solutions, secure networks, defence electronics, sensors and avionics, as well as related services.

EADS is a global leader in aerospace, defence and related services. In 2005, EADS generated revenues of € 34.2 billion and employed a workforce of about 113,000.

ASSERT Project — Ada in the European space industry

http://www.adacore.com/2006/09/05/assert-project-adopts-ada-2005/

Tuesday September 5, 2006

ASSERT Project Adopts Ada 2005

New Ada standard improves embedded real-time development.

Ada EUROPE, PORTO, 5 June 2006 – ASSERT, the EU-sponsored project to improve the development of critical embedded real-time systems today announced that it will adopt the Ada 2005 language and AdaCore's open-source GNAT development environment.

ASSERT (Automated proof-based System and Software Engineering for Real-Time) will use Ada 2005 to develop a run-time platform and integration framework and re-usable components that can be adopted into Aerospace and other high-integrity sectors to aid the development of real-time, safety critical systems. Begun in 2004, the €15 million project will begin its 18 month final development phase in June July-August 2006.

A joint industrial and academic consortium, ASSERT brings together leading players from the European space industry, SMEs and research organisations to improve the system and software development process for critical embedded real-time systems. Part of this process is to create building blocks of open source code covering common areas along with a run-time platform and integration framework for them to operate within. The UPM (the Technical University of Madrid) Madrid and ENST Paris (Ecole Ingenieur Télécom Paris — ENST Ecole nationale supérieure des télécommunications) are developing the run-time platform framework while Alcatel, EADS, Astrium France and other leading industrial partners will create the first building block application level components for it. These will be developed in a variety of programming languages that will be weaved and integrated for execution on the run-time platform using into the framework through Ada 2005 "capsules" interfaces to and C/C++interfaces. The components will then be made available via open repositories, allowing them to be used by developers to speed up safety-critical development.

"Developing critical real-time embedded systems is central to the continued development of the European aerospace

industries," commented Juan Antonio de la Puente, professor, UPM. "Ada 2005 was the only language that provided us with a reliable, open-source environment that will enable us to create the framework for a new generation of safety critical systems. Working with AdaCore was the natural choice due to its knowledge and advanced implementation of Ada 2005 in products."

The Ada programming language is designed specifically for large, long-lived applications where reliability, efficiency and safety are vital. Created under the auspices of the International Organization for Standardization (ISO), Ada 2005, the latest version of the language, was ratified last year. It introduces significant enhancements in many areas including Object-Oriented Programming, interfacing with other languages (most notably Java), software architectural design, real-time systems, and predefined libraries. It offers improved support for high-integrity applications, including the standardization of the Ravenscar profile for certifiable concurrent programs. Ada 2005 represents the first major upgrade of the Ada language in ten years.

Ada 2005 was chosen by ASSERT due to its seamless integration of budget monitoring, timing events, priority-specific dispatching, execution time monitoring, 'limited with clauses', new interfaces and compile and run-time support for the Ravenscar Profile.

"ASSERT's choice of Ada 2005 and GNAT demonstrates that it provides the perfect solution for real-time embedded system development, combining safety-critical features with the benefits of an open source development environment," said Cyrille Comar, managing director, AdaCore. "The combination of Ada 2005, ASSERT and AdaCore aims to make significant progress in driving forward European embedded system development."

ASSERT developers working on the both the run-time framework and pilot applications will be using Ada 2005, with an estimated 33,000 lines of code predicted to be written, making it the largest current use of the latest version of the language.

AdaCore has been closely involved with the Ada language since its inception and its GNAT development environment combines market leading technology, including Ada 2005, with an expert support system to provide a natural solution where efficient and reliable code is critical.

About ASSERT

ASSERT (Automated proof based System and Software Engineering for Real-Time) is an integrated project (IP) partially funded by the European Commission under the Information Society Technology (IST) priority within the 6th Framework Programme (IST-FP6-2004 004033). The project addresses the strategic objective of the Embedded Systems sector. ASSERT brings together an important consortium, coordinated by the European Space Agency, which includes leading actors in the European Space industry, SMEs and research organisations with the determination, skills and critical mass to create cross-industry consensus and to take the project outputs through to standardization.

ASSERT is a 2-phased project spanning from Q3 2004 until Q4 2007. Phase 1 has laid out the conceptual and methodological foundations on which the development work in Phase 2 will base. Phase 2 will demonstrate the operation of a tool-assisted model-driven reuse-oriented property-preserving development process in the production of 3 pilot applications consisting of property-endowed building blocks controlledly instantiated and integrated by property-preserving interfaces and and bindings, and executed on a property-preserving run-time environment.

For more information about ASSERT please contact:

Eric Conquet
European Space Agency
eric.conquet@esa.int

AdaCore — Boeing's Real-time Simulation Systems

http://www.adacore.com/2006/07/25/gnat-pro-development-environment-to-support-boeing%e2%80%99s-real-time-simulation-systems/

Tuesday July 25, 2006

GNAT Pro to Support Boeing's Real-time Simulation Systems

AdaCore's GNAT Pro Development Environment to Support Boeing's Real-time Simulation Systems

NEW YORK, July 25, 2006 – AdaCore today announced that it has been awarded a contract to provide its flagship GNAT Pro Ada toolset and development environment to The Boeing Company's Training Systems & Services (TSS) division in St. Louis, MO, where it will be used for the development of Ada flight simulation code. AdaCore has been supplying Boeing with Ada development tools for real-time aircrew training simulation systems since 2003, including the C-17, the U.S. Army AH-64D Apache Longbow (an all-Ada aircraft), and the F-15E Rapid Prototyping System (RPS). As part of the contract, AdaCore will also be supplying GNAT Pro for the development of training systems for custom Apache aircraft that will be delivered to Japan and Kuwait. TSS specifically utilizes AdaCore's GNAT Pro, including the GNAT Programming Studio (GPS) Integrated Development Environment, for software development, testing and debugging on x86 Linux and x86 Windows platforms.

"Ada has been used successfully in Boeing programs for a number of years, both for training device software development and for adaptation of operational flight programs used in the flight hardware of the aircraft being simulated," said Robert Dewar, CEO of AdaCore. "The use of common, proven system components, such as our GNAT Pro Ada environment, means the technology can be easily integrated into a wide variety of systems at a significant cost savings, which maximizes operational readiness and reduces total ownership costs."

About Boeing's Training Systems & Services Division

The Boeing Company's Training Systems and Services (TSS), headquartered in St. Louis, Mo., provides the full range of military and government training and mission planning system solutions for domestic and international customers. TSS consists of over 2,400 employees, has operations at six major sites, and provides training support at numerous sites located throughout the world. TSS' business scope, which is based on years of expertise gained in the design and development of more than 150 trainers for 24 different aircraft, encompasses fully integrated training systems, as well as comprehensive services that include instructors, courseware developers, logistics support and mission planning systems.

Indirect Information on Ada Usage

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

Mariland, United States

From: Atul Bhardwaj <atul.bhardwaj@mastech.com>
Subject: Unix and Ada Programmer/Analyst with CSC @ Rockville MD
Date: 25 Aug 2006 14:20:33 -0400
Newsgroups: comp.lang.ada

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.

Brussels, Belgium

[…] 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 personalized 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.

Florida, United States

Need two (2) VERY experienced Ada contract programmers for an 18+ month assignment to develop/test embedded (PowerPC) Ada software on a GPS/INS product:

* Must be an expert in Ada

* Must have extensive experience developing military-rated software

* Experience with GPS and Inertial navigation software a plus

* Experience with RavenAda toolset a plus.

No telecommuting allowed.

California, United States

Position is for real-time, avionics system/software engineering […] Work is in the software development organization defining requirements, developing interface control documents, and performing system/software integration and testing. The software being developed is real-time avionics software. Work may involve displays, operating systems, or flight controls.

Required Skills: 1) Experience across all software development phases including requirements analysis, top level and detailed design, code, integration and test. 2) Experience in software requirements definition and interface requirements analysis including writing SRS and ICD documents. 3) Experience with testing avionics or similar architectures including designing test cases, writing test scripts, and conducting tests using sophisticated test/simulation environments. 4) Ability to work independently and solve difficult technical problems with minimal assistance. Must be able to focus and solve problems while also multi-tasking on more than one task. Must demonstrate persistence, patience, and flexibility in problem solving. 5) Good communication skills, ability to work in team environment, ability to work constructively with other strong personalities.

Desired Skills: 1) Experience writing automated test scripts. 2) User knowledge of configuration management processes and CM tools such as ClearCase. 3) User knowledge of DOORS. 4) Experience with interface protocols such as 1553, ARINC 429, RS-232, RS488, or TCP/IP

Ada in Context

Ada 83 Reference Manual

From: Jeffrey R. Carter <spam.not.jrcarter@acm.not.spam.org>
Subject: Re: 83 LRM
Date: Mon, 17 Jul 2006 20:31:42 GMT
Newsgroups: comp.lang.ada

> Can someone tell me where I can find a PDF version of the '83 LRM?

There's an HTML version at adaic.org; I'm not aware of a PDF version.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: 83 LRM
Date: Mon, 17 Jul 2006 19:33:16 -0500
Newsgroups: comp.lang.ada

To my knowledge, the 83 RM was produced directly to paper. (It long predates things like PDF!). I don't think the original Scribe files exist anymore, either. The HTML version was created by hand by AJPO (which is why it is full of markup errors; I can't vouch for the accuracy of the text, either).

The only reason that there is a PDF of the Ada 95 version is that I made one to test the new tools at the start of the project that created the (2001) Consolidated RM. It has a few formatting errors that have been discovered more recently (one just a month ago), so even it can't be considered "official".

The PDF of the 2001 Consolidated RM is, OTOH, the same as what was printed by Springer, and I expect the same will be true for the 2005 version when that is finished up. (Keep in mind the version currently available is the last draft, and it is possible that some changes will be made in the final version. There definitely will be some changes in Annex P and the index (not normative, of course).

From: "Stuart" <stuart@0.0>
Subject: Re: 83 LRM
Date: Tue, 18 Jul 2006 08:11:06 +0100
Newsgroups: comp.lang.ada

Digital (aka DEC, now HP) produced a postscript version of their DEC Ada Language Reference Manual which describes itself as

"… the Digital-supplemented text of ANSI/MIL-STD-1815A-1983"

Having had a quick search there seems to be a PDF version at:

http://h71000.www7.hp.com/commercial/ada/ada_lrm.pdf

This might be helpful.

From: "Charlie McCutcheon" <charlie.mccutcheon@hp.com>
Subject: Re: 83 LRM
Date: Thu, 27 Jul 2006 14:35:00 GMT
Organization: Hewlett-Packard Company
Newsgroups: comp.lang.ada

> Of course it has all the Dec… — HP Ada extras in green writing.

And the little numbers in the margins for the black text are supposed to be the original paragraph numbers to the original Ada 83 language standard…

Ada for VMS

From: Kilgallen@SpamCop.net (Larry Kilgallen)
Subject: Re: Is Ada on VAX/VMS Ada 95
Date: 26 Jun 2006 14:36:58 -0500
Newsgroups: comp.lang.ada

> What are the differences of VAX Ada and Ada 95?

VAX Ada conforms to the Ada 83 standard. […]

VAX Ada became DEC Ada became Compaq Ada became HP Ada and runs on either VAX or Alpha, implementing Ada 83.

If you want an Ada 95 implementation on the VMS operating system, use GNAT Ada 95 for Alpha VMS. It has many of the VMS-specific capabilities of the original VAX Ada.

From: Keith Thompson <kst-u@mib.org>
Date: Tue, 27 Jun 2006 02:12:35 GMT
Subject: Re: Is Ada on VAX/VMS Ada 95
Newsgroups: comp.lang.ada

> One major reason why there is no Ada 95 implementation on VAX is that Ada 95 requires IEEE floating point semantics while the VAX hardware provides VAX floating point semantics.

I don't believe that's correct. The floating-point model is intended to be

flexible enough to cover any existing hardware that implements a mantissa-exponent model (IEEE, VAX, IBM, etc.). See section G.2.1.

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: Re: Is Ada on VAX/VMS Ada 95
Date: Tue, 27 Jun 2006 18:57:03 +0200
Newsgroups: comp.lang.ada

> The GNAT Ada 95 is not available for VAX at all. It is available for Alpha VMS, which runs on the Alpha hardware.

And IA64 OpenVMS — just saw it today on GNAT Tracker.

The GNU Ada project has got one — but it is horribly out-dated.

> That said, some shops imprecisely refer to any machine running the VMS operating system as being a "VAX". I figure they are not the type of shops to choose strongly typed languages :-)

I would think that all VAXes are now in a cosy little retirement home and the OP was just new to the wonderfull world of OpenVMS.

Global variables

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Subject: Re: [SPARK] Code safety and information hiding
Date: 18 Aug 2006 19:02:20 -0400
Newsgroups: comp.lang.ada

> I really don't like the use of the term "global" for state variables. True global variables (in package specs) should be illegal; state variables are a different cup of fish and should not be tainted by the "global" label.

I think the terms "global" and "local" are most useful if they are relative to something. A variable can be more global or less global. The Ada RM agrees with that (not surprising, since I wrote that part ;-)). See quote below.

Jeff wants to consider package-spec variables particularly evil. OK, but there are global variables even more global than that. The Registry in windows is an obnoxiously global variable. And environment variables under many operating systems are obnoxiously global. I suppose disk files are global, although I don't usually find them obnoxious.

I don't agree that package-spec variables should be illegal. Such a rule accomplishes nothing, since the programmer can just have a getter and a setter, which is just as evil. In some rare cases, a package-spec variable makes sense, and in those cases, the getter/setter method is no better (just more verbose).

So I like the SPARK syntax for "global". It means so-and-so variable is global to this procedure, not necessarily global to the whole program, or global to that part of the program that with's certain package, or global to the whole world.

AARM-8.1 says:

14 [{local to} A declaration is local to a declarative region if the declaration occurs immediately within the declarative region.] [An entity is local to a declarative region if the entity is declared by a declaration that is local to the declarative region.]

14.a Ramification: "Occurs immediately within" and "local to" are synonyms (when referring to declarations).

14.b Thus, "local to" applies to both declarations and entities, whereas "occurs immediately within" only applies to declarations. We use this term only informally; for cases where precision is required, we use the term "occurs immediately within", since it is less likely to cause confusion.

15 {global to} A declaration is global to a declarative region if the declaration occurs immediately within another declarative region that encloses the declarative region. An entity is global to a declarative region if the entity is declared by a declaration that is global to the declarative region.

From: Peter Amey <peter.amey@praxis-cs.co.uk>
Subject: Re: [SPARK] Code safety and information hiding
Date: Wed, 23 Aug 2006 10:44:54 +0100
Newsgroups: comp.lang.ada

That's certainly the design decision we made with SPARK. The term global is always relative to the place where you are asking the question. If an entity is declared outside the declarative part of the place from which you are looking it is global otherwise it is local. A local variable of a subprogram may well be a global to a nested subprogram for example. SPARK's global annotation eliminates "hole in scope" issues (as well as doing other useful things).

SPARK doesn't prohibit package-spec variables (actually it did in the very early days but a large and influential customer made us change our minds); however, it does nag you in various ways if you make use of them. Abstract own variables and refinement clauses are a much better solution!

From: Peter Amey <peter.amey@praxis-cs.co.uk>
Subject: Re: [SPARK] Code safety and information hiding
Date: Thu, 24 Aug 2006 11:55:25 +0100
Newsgroups: comp.lang.ada

> I think it would be better if you'd educated your customer on how to use that better solution, and kept SPARK as it was.

Now we probably would, then (1991?) we were very small and we _really_ needed that customer!

Our nagging really is persistent though and I don't think any current SPARK projects continue to abuse own variable visibility.

From: Jeffrey R. Carter <spam.not.jrcarter@acm.not.spam.org>
Subject: Re: [SPARK] Code safety and information hiding
Date: Sat, 19 Aug 2006 05:40:39 GMT
Newsgroups: comp.lang.ada

Designs based around global (package spec) variables are clearly done by people who are not competent to be designing SW. Such people can always find others ways to create SW that's just as bad. But such a person might not immediately think in terms of getter/setter operations if denied the use of global variables, so not allowing them might provide a little pressure towards better designs.

The real solution, recognizing that all SW people are not equal, doesn't seem likely to happen any time soon.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sat, 19 Aug 2006 05:49:14 -0400
Subject: Re: [SPARK] Code safety and information hiding
Newsgroups: comp.lang.ada

I have an application called Goddard Dynamic Simulator (GDS); it is used to test flight software and hardware for NASA Goddard (yes, I am a rocket scientist). It has 90,000 lines of code, mostly Ada, some C and VHDL.

The design uses information hiding, abstraction, etc; all the good features of a good program, made easy by using Ada 2005.

However, there are some global variables. I thought long and hard about each one, and decided they are the best solution to the problem.

One important global variable is the root of the symbol table. GDS consists of many modules, that use the symbol table to communicate. One module models a star tracker, another a spacecraft (which contains the star tracker), another thrusters, etc. In addition, any symbol can be written by the user, or displayed to the user.

This design has evolved thru several iterations of similar systems. They all have a global symbol table.

In addition, there are global variables that indicate global information about mode:

```
type Distribute_Mode_Type is
   (Master_Mode, Remote_Mode,
    Single_Mode);
```

```
Distribute_Mode :
  Distribute_Mode_Type :=
    Single_Mode;
```

This indicates whether GDS is running on a single computer, or on several. This makes a difference in many places in the code.

```
Ignore_Hardware_Errors : Boolean
  := False;
--  Set True for unit tests when
--+ hardware is not present or should
--+ be ignored.
```

Unit tests are important, and this feature is the simplest way to support running them without hardware, in a system that normally talks to hardware.

```
Main_Config : SAL.Config_Files.
  Configuration_Type;
```

More information that the entire system needs.

I claim to be competent to design SW; I have 20 years of experience, and at least 10 monetary awards for outstanding performance, together with high praise from my (internal) customers, to prove that. My system has a few global variables because they are the right solution to the problem.

You have not said why getter/setter would be better for these variables. I agree with Robert; they are just more verbose.

Hmm. You could try to impose control over what parts of the system are "allowed" to write the variables, as opposed to reading them. But that would require an elaborate system of IDs for various parts of the system (which does not otherwise exist); definitely not worth it.

I'll make a counter claim; people who claim global variables are _always_ bad should not be designing large complex systems; the systems will be more complex than necessary, which is definitely a Bad Thing.

While it is true that all SW people are not equal, I don't see what that has to do with the issue of global variables, nor do I see what problem recognizing that might be a solution to.

From: Jeffrey R. Carter <spam.not.jrcarter@acm.not.spam.org>
Subject: Re: [SPARK] Code safety and information hiding
Date: Sun, 20 Aug 2006 03:52:27 GMT
Newsgroups: comp.lang.ada

Some is probably better than [lot of global variables]. With "some" they might be well documented. As I've said, I've seen large, safety-critical systems designed around thousands of undocumented global variables. I think it would be faster and easier to redesign and reimplement these systems than to understand them well enough to safely make changes to a small part of them.

I've often had similar thoughts about systems in which simple boolean options exist which may be changed at any time. If it's a sequential system, Boolean variables seem sufficient. For a concurrent system, making them atomic should be enough, though I would prefer a protected object. On the other hand, I don't want to seem to give permission for modifiers of the system to create global variables that are more complex or have more restricted situations when they may be accessed.

So, as a modifier of the system, without reading and understanding all parts of the system that may access this variable, it's OK for me to write to this variable whenever it's convenient?

It must be reasonable and meaningful for me to change the SW to write [the Distribute_Mode variable] whenever it suits me. If that's not the case, then I claim this should not be a variable. Instead of being able to write

```
X.Distribute_Mode :=
  X.Master_Mode;
```

I should have to call something like

```
function
Request_Distribute_Mode_Change
  (New_Mode :
    Distribute_Mode_Type)
  return Boolean;

--  Request mode change to
--+ New_Mode.
--+ Returns True if the mode change
--+ is legal; False otherwise.
```

All the details of when a mode change may occur should be encapsulated with the current mode value.

On the 3rd hand, systems generally are either distributed or not, and that doesn't change during execution. In that case, this is essentially a constant. If the actual mode is determined at run time during start up, then probably the value should be encapsulated with the logic that determines it, and the resulting value be made available to the rest of the system through a function.

In this latter case, there may also be a race condition. What if one part of the system reads this and gets the default value before its actual value is determined? It's probably better to control that locally than otherwise. […]

Who modifies the system? Add one undocumented global variable to a system, and you generally increase the coupling and decrease the cohesion. You understand the system, but after you're hit by a truck, how easy is it for someone who doesn't have any experience with the system to figure out how it works?

I don't think [getter/setter] are better. I said that if global variables were not available, those who would try to base their designs around them might not think of using such operations to mimic them.

You might simply document who may write and who may read such variables. That would be a big improvement in the global-centric systems I've seen. Of course, that goes against the basic Ada philosophy that packages should not be concerned with who their clients are. That's something that's not always achievable in large, real SW systems. There are sometimes operations that are documented as "used only by X and Y to achieve Z". Such things should generally be private to some subsystem.

In general, values should be encapsulated with the logic that generates or modifies them, and systems that are designed that way have better coupling and cohesion than systems that aren't. Systems designed around global variables are an extreme example of the latter. If your SW determines a value in one place and stores it in another, that's an indication that you may need to rethink your design. If the value is determined in multiple places, that's almost always a red flag.

Now, I've worked on systems with physical redundancy, so that one processor is the currently active one, and the others shadow it and are prepared to take over if it fails. I've worked on such systems where the determination of which is the active processor is made by another part of the system, so it is necessary to distribute that value to the processors. In that case, the SW pretends that it determines the value; for example, there's a module that encapsulates the value and provides access to it to other parts of the SW. There isn't a procedure Set for the value; rather, the module has an operation that receives and processes the message that changes the value.

I have over 30 yrs of professional SW development experience, and I still find myself doing things I know I shouldn't. Hopefully I catch them most of the time in the important cases. But I've seen people with decades of experience who are not competent to design any meaningful SW. They're among that large proportion of developers who will never be more than coders. Customers are happy with anything that works and they can afford; I've seen customer satisfaction with very poorly designed SW.

There may be cases where global variables are the correct design approach, but I have yet to see one.

Once you recognize that [all SW people are not equal], you don't allow the coders to design SW. That gets rid of most poor design, including designs with large numbers of undocumented global variables.

As I said above, I have many designs that are not as good as they could be, and some of them are freely available for anyone to look at. My primary problem is with designs where the global variable is the determining feature, with large numbers of them, and the fact that in over 30 yrs I've never seen a case where global variables were the best approach. GDS is probably not in that category. I'm not saying the GDS SW is poorly designed (I've seen enough of your work to suspect it probably isn't), but I suspect that the design could be improved and that the improvement would eliminate the global variables. Whether the improvement in design quality would be worth the effort to achieve it now, I can't say. At the time of initial design, it almost certainly was.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: Re: [SPARK] Code safety and information hiding
Date: Sun, 20 Aug 2006 15:06:04 -0400
Newsgroups: comp.lang.ada

[…] "Rocket scientists" are actually typically engineers. The original term was for the people who made the first large rockets actually fly; they were certainly engineers. […]

In my system, as you alluded to above, the use of each global variable is well documented; programmers are expected to follow the rules in the documentation.

Of course, every time I add a new programmer to the team, they discover new ways to (mis-)use the globals that were not covered by the documentation :).

You want the code to enforce the design rules. That's not possible in general. Ada allows enforcing some design rules, but not ones at this level.

In fact, for this particular variable, the rule is "only the environment task may write to this variable, and only before any other tasks are started".

People modifying the code must understand the design.

I have been tempted to write some ASIS programs to enforce more of the design rules. One that people keep breaking is "modules must not write to their own Input symbols". But this would be a complex program (worse than Auto_Text_IO, for example), so I haven't done it yet.

Hmm. Now that I think about it, it would be easy to implement a run-time check for that rule. But it would be a waste of time in the production system. Might still be worth it; I have other run-time checks for similar rules (in particular, "symbols must be written before being read") that are hard to check otherwise.

> On the 3rd hand, systems generally are either distributed or not, and that doesn't change during execution. In that case, this is essentially a constant. If the actual mode is determined at run time during start up, then probably the value should be encapsulated with the logic that determines it, and the resulting value be made available to the rest of the system through a function.

That would be essentially a globally visible getter/setter; the getter must be globally visible, but the environment task which calls the setter should not be. I don't see any way to enforce the rule of which task is allowed to set the variable at run time.

In practice, the rule of how this variable is set has never been broken. […]

During any particular run of the full production system, [Ignore_Hardware_Errors] is a constant; usually False but occasionally True (if some important hardware is absent). Thus it is set by the environment task at start-up, just like Distribute_Mode.

During runs of many unit tests together (under an AUnit test harness), this value changes between False and True. Some unit tests use hardware emulation code, and can check for hardware errors. Other unit tests have no emulation code, and must ignore them.

[…] Add one undocumented function/procedure/package/task to the system, and you have a problem.

The key notion here is "documentation"; any system must be documented, so that maintainers understand it before modifying it.

When one of my team members or customers asks a question about how to use the system, I try to look in the documentation to answer it first. If the answer isn't there, I fix the documentation as the answer to the question.

Of course, when there's a crunch on, questions get answered more directly. So the documentation isn't perfect. But it is intended to be adequate. […]

"Large numbers" and "undocumented" are the key here, and as I pointed out above, that description applies to other aspects of the software as well; large numbers of small functions, or large numbers of threads, are just as bad (possibly worse) as large numbers of global variables.

[…] At the time of the initial design, I decided that eliminating the global variables was more costly than the risk of them being misused.

It does depend on the quality of the maintenance team; whether they truly understand the system before modifying it. But someone who doesn't understand a system will still be tempted to "just add a global variable to fix this". Which is why adding complexity just for the sake of reducing global variables is not worth it; it makes the system that much harder to understand.

Opaque Types

From: claude.simon@equipement.gouv.fr
Subject: Re: How to hide type internals
Date: 30 Jun 2006 04:19:11 -0700
Newsgroups: comp.lang.ada

> I would like to write a package that exports functions using a special type, without showing the details in the spec (even not in the private part).
> In Modula-2 one can declare opaque types (which mostly are pointers but also can be any type of word size).

```
package Keys is
  type Key is private;
  function Getkey return Key;
-- …
private
  type Imp_Key; --   deferred type to
                --+ the body
  type Key is access Imp_Key;
end Keys;

package body Keys is
  type Imp_Key is new Integer;
  -- whatever type
  function Getkey return Key is
    The_Key : Key := new Imp_Key;
  begin
    return The_Key;
  end GetKey;
end Keys;
```

From: Alex R. Mosteo <devnull@mailinator.com>
Subject: Re: How to hide type internals
Date: Fri, 30 Jun 2006 13:26:11 +0200
Newsgroups: comp.lang.ada

I've done this in occasions where I want to have a multi-platform package, so switching bodies is enough, keeping the same spec file. Like this:

```
package Blah is
  type External is private;
  -- Operations on external
private
  type Internal; -- forward declaration
  type Internal_Acess is access all
    Internal;
  type External is record
    Ptr : Internal_Access;
  end record;
end;
```

And then you define Internal in the body.

With Ada 2005 you can even get rid of Internal_Access and simply have

```
type External is record
  Ptr : access Internal;
end record;
```

Of course you can have reference counting, deep copying or whatever making External a Controlled type, so you can have a proper abstraction (This will

be probably a requisite for initialization anyways…)

From: Stephen Leake <Stephe.Leake@nasa.gov>
Date: Sun, 02 Jul 2006 11:23:02 -0400
Subject: Re: How to hide type internals
Newsgroups: comp.lang.ada

Others have shown how to do this.

But I'd like to understand _why_ you want to do this.

If it is just to explore the capabilities of the language, that's fine.

But you should also understand the compiler issues involved. The reason Ada provides a private part in package specs is to provide information to the compiler to allow it to generate more efficient code (not using pointers, for example).

One downside of providing information in the spec is more recompilation when the private part changes.

From: Gerd <GerdM.O@t-online.de>
Subject: Re: How to hide type internals
Date: 4 Jul 2006 04:11:52 -0700
Newsgroups: comp.lang.ada

There are at least two reasons:

If the definition is left to the body, a change of the definition needs only a recompilation of the body. If the full type is declared within the spec, you would have to recompile .all. packages that "with" this spec. Otherwise it would be enough to provide a compiled object-file with the spec.

First this is a question of compile time (large system), second this would give the contractor a look into your internals. The first is a simple thing of money, the second is a security decision (e.g. if the "user" of this package has no clearance).

Lexical Analysis

From: Keith Thompson <kst-u@mib.org>
Subject: Re: lexical ambiguity
Date: Fri, 02 Jun 2006 23:27:04 GMT
Newsgroups: comp.lang.ada

> I'm doing a lexical analysis of Ada using Lex as part of a student project. The highlight is on using Lex, not on the programming language of Ada and I'm not familiar with using Ada. So what I would like to find out is if there is any lexical ambiguity in Ada (like the ambiguity in C with the unary and binary plus and minus). Thanks in advance…

I suppose it depends on what you mean by "lexical ambiguity".

Strictly speaking, there are no grammatical ambiguities in either language. There are plenty of things that look like ambiguities, but they're all resolved by the rules of the language.

In C, for example, this:

```
x+++++y
```

looks like it could be parsed as

```
x ++ + ++ y
```

which would be a legal expression, but in fact it's tokenized as

```
x ++ ++ + y
```

which results in a syntax error. (C's typedef names do cause some interesting lexical problems, but that's another topic.)

Ada, like, C, has unary and binary "+" and "-" operators, but each operator is easily identified based on the syntactic context in which it appears. One well-known case of a near ambiguity is:

```
Character'('x')
```

If Ada followed C's "maximal munch" rule, this would be tokenized as

```
Character '(' x '…
```

leading to a syntax error; instead, it's tokenized as:

```
Character ' ( 'x' )
```

So, there are no real ambiguities in either language, but each uses different rules to resolve things that would otherwise have been ambiguous.

From: Frank J. Lhota <FrankLho@rcn.com>
Subject: Re: lexical ambiguity
Date: Fri, 2 Jun 2006 18:35:55 -0400
Newsgroups: comp.lang.ada

The biggest lexical issue with Ada is the multiple uses of the single quote:

- Single quotes surround character literals (e.g. 'A'),

- prefix attributes (for example List'First), and

- are used in [qualified expressions], such as Rational'(Num =>1, Demom => 2).

From: Jeffrey R. Carter <jrcarter@acm.org>
Subject: Re: lexical ambiguity
Date: Mon, 05 Jun 2006 01:36:19 GMT
Newsgroups: comp.lang.ada

> Make sure that your lexer can handle the following expression properly:
> `Foo'(',','','','' … )`

Clearly you have an evil mind :)

From: Keith Thompson <kst-u@mib.org>
Subject: Re: lexical ambiguity
Date: Mon, 05 Jun 2006 20:27:11 GMT
Newsgroups: comp.lang.ada

> Well, there is a good reason to consider this worst case scenario. I have seen quick and dirty Ada lexers that try to determine if a single quote starts a character literal by looking ahead 2 character. As this scenario shows, this approach is not guaranteed to work.

If I recall correctly, it's sufficient to remember what the previous token was. A character literal cannot follow an identifier.

I think that might break down if an implementation chooses to define an attribute with a single-character name, but I don't remember the details; presumably no implementation will actually do this.

From: Jeffrey R. Carter <spam.not.jrcarter@acm.not.spam.org>
Subject: Re: lexical ambiguity
Date: Mon, 05 Jun 2006 22:11:37 GMT
Newsgroups: comp.lang.ada

Right, so it must be either an attribute, a qualified expression, or an error. An attribute must be an identifier, so it can't be an attribute, so it's either a qualified expression or an error. In this case, it's an error, since you can't have "…" as part of an aggregate :)

From: M E Leypold <kontakt@m-e-leypold.de>
Subject: Re: lexical ambiguity
Date: 06 Jun 2006 13:38:06 +0200
Newsgroups: comp.lang.ada

> Though the previous token shouldn't be a reserved word, as in
> `if'('="-"("="('='=',',',','='))`

Or

```
return'a';
```

So now (question to all): Is the following rule enough?

- """ is the beginning of a character literal if the token before

""" has not been an identifier (reserved words not counted as identifier in this case).

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Subject: Re: lexical ambiguity
Date: 07 Jun 2006 10:49:31 -0400
Newsgroups: comp.lang.ada

Not quite:

```
function F(X: Integer) return String;
Length: constant Natural :=
F(123)'Length;
Y: access T'Class := …;
Z: access T2'Class := Y.all'Access;
```

For reserved words, I think you have to study the grammar, and determine which ones can precede a tick mark.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: lexical ambiguity
Date: Wed, 7 Jun 2006 11:02:51 +0200
Newsgroups: comp.lang.ada

It does not differ from the case of +/-. In the infix context, i.e. after an operand (whatever it might be), ' is an infix operation as well as +/-. In the prefix context, where an operand is expected ' introduces a character literal (=operand), +/- do an unary prefix operation.

Your rule is wrong: 'A' and 'B'. "and" is a reserved word. Then of course "…" comments should be parsed before.

Which gives you a nice vicious circle around ' " ' and " ' ". (:-))

The bottom line: parsing has state.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Subject: Re: lexical ambiguity
Date: 08 Jun 2006 17:30:53 -0400
Newsgroups: comp.lang.ada

To determine whether a single quote begins a character literal versus a tick, it is sufficient to look back one token. Some tokens can be followed by a tick, some by a char_lit, and some by neither. None can be followed by both. It's fairly straightforward to study the grammar and determine which are which. Or look at the GNAT sources.

It might be wise to include a sentinel token at the start of the token stream (Begin_File_Token or whatever), just in case ' comes first (that would be illegal, but you don't want to crash on it).

It can all be done in the lexer, with no feedback from the parser — the lexer just needs to keep track of the previous token, and check it when it sees a single quote. Lookahead will get you in trouble; look-back is the better answer here.

> The ensuing discussions leaves me more and more doubtful: Can lexical analysis (grouping characters to tokens and grammatical analysis (building a parse tree from a token sequence) be separated cleanly in Ada?

Yes. The look-back is localized to the lexer (which is not "clean", but at least it's localized (separated from the parser)).

> My first approach would have been (no I'm not implementing an Ada parser, but since compiler construction has been a favorite subject of> me for a number of years, I'm a bit curious about the position of Ada in all this) — now: My first approach would have been, to write a lexer with a minimal amount of state. It would shift into collect-string state when encountering a "" (I mean a double quote :-) and into especially into maybe-now-comes-a-character-literal state at certain points. My first take was that the "certain points" are always after identifiers. In view of the case quoted above (F(123)'Length) I could amend this rule by adding ')' to the certain points.

Right. But you have to study the grammar to know which tokens have this property. It's not that big of a deal.

> But now things become rather ad-hoc. Well — as I said, that it's just curiosity driving me, so I'm not going now to examine the RM not I'm going to reverse engineer GNAT to find out how it is done in reality.
> But if anyone in c.l.a. has the answer to the following questions, I'd be eternally grateful. Well, grateful, anyway. :-)
> - Is it possible (for Ada parsers) to separate lexical analysis and grammatical analysis into separate phases without tricky feedback from parser to lexer, possibly by using a lexer with a finite amount of states.

Yes. Just a tiny bit of state — the previous token. The lexer writer needs to understand the grammar, but the lexer does not need to understand the parser.

> - What is the complete rule for deciding when the next token might be a character literal. Or is that undecidable by just looking on past input (i.e. using lexer state)?

It is decidable by looking at the previous token. I forget the exact rule, but it can be deduced easily from the grammar.

> BTW: The "evil" case
> if'("-"("="('=',','=',')) is not parsed OK by syntax highlighting in Emacs ada-mode (I wouldn't have expected it, actually). The rule there seems to be my incomplete rule without the reserved words exception.
> Everything falls magically into place if a " " is inserted immediately after "if".

I'm not surprised. Emacs ada-mode uses some ad-hoc technique that doesn't always work properly. Anyway, Emacs is trying to parse bits and pieces of things without seeing the whole file, and that's a whole 'nother thing. It is certainly easy to parse the above "evil" thing properly, but not necessarily if you start in the middle of it.

Elaboration Issues

From: Alex R. Mosteo <devnull@mailinator.com>
Subject: Elaboration worries
Date: Wed, 21 Jun 2006 14:33:06 +0200
Newsgroups: comp.lang.ada

I'm revisiting this topic in hope of enlightenment by someone that really understand the dark magic behind elaboration. I've read the relevant sections of the ARM but I can just grasp a more-or-less depth understanding, of which details fade away with time.

From past discussions and from reading an old article of Mr. Dewar, my rule of thumb is that you should

a) make your package Pure.

b) if not possible, make it Preelaborate.

c) if not possible, put an Elaborate_Body in the spec.

I was happy following this rule, but just recently I've started to experiment with the -gnatwl switch, that warns when a "Elaborate_All" is needed. I have two questions related with this.

The first one is that I don't really grasp why a "Elaborate_All" could be needed if one strictly follows the tree rules above.

The second one is this: by my observation of the GNAT warnings, it seems that the requirement for "Elaborate_All" is a "server side" one. However, I must use the Elaborate_All in all "client side" units. This is because contrarily to the abc) rule pragmas, that can refer to the package where they appear, Elaborate_All can not.

Example: Package A causes a warning for Elaborate_All in all packages that use A. I can't do nothing at A to remove the warning, but to put an Elaborate_All(A) in all client packages. This seems strange to me.

A final note is that this second situation happens to me mostly with generic units.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Elaboration worries
Date: Wed, 21 Jun 2006 15:34:12 -0500
Newsgroups: comp.lang.ada

Unfortunately, this rule of thumb is wrong (you probably got it from reading the old article). Adacore later discovered that having Elaborate_Body in a spec does not eliminate elaboration problems, and their modern versions of GNAT don't make any recommendation for this. We were bitten rather badly by this with Claw (which followed your list of recommendations quite closely). I forget the exact reason that Elaborate_Body doesn't work, but the effect is that you can't count on it to eliminate elaboration problems completely.

Effectively, if you do much of anything at elaboration time, you'll need Elaborate_All in the clients for any packages that aren't at least Preelaborate. There is nothing whatsoever that can be done for the service packages that prevents that. (It's really a flaw in Ada, but one that cannot be fixed without radical surgery and incompatibilities — not an option these days).

My personal rule of thumb is:

(a) make your package Preelaborate if possible (Pure is so limited that no real packages ever qualify) — but this is usually impossible because I/O and Calendar aren't Preelaborate. Which means that you can't trace or log a Preelaborate package (well, there *is* one way to do it, but it adds runtime overhead);

(b) if not (a), try to avoid doing anything at elaboration time other than use language-defined packages;

(c) if not (b), then you have to add Elaborate_All for anything used at elaboration time.

(c) usually happens when you have generic instantiations at the library level, or you declare controlled objects at the library level. The latter can be avoided, the former obviously can't.

I've long since given up running code (the 'begin' part of a package body) at elaboration time; there always seems to be

some reason that you have initialization dependencies that aren't encoded in the elaboration order (for instance, the need to load parameters from the registry or a configuration file before starting a subsystem). I use appropriate initialization routines (and often checks for calls to other routines in the package that the initialization has been properly called).

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Elaboration worries
Date: Thu, 22 Jun 2006 18:06:02 -0500
Newsgroups: comp.lang.ada

> There are cases where (b) is quite impractical; for example, when using the Ravenscar profile, tasks cannot be created dynamically and will be started at elaboration time. Using a "start" protected object can be used as a workaround to limit their execution until after the main program has been launched.

Which is why I said "try to avoid", not just "avoid". It's hard to avoid library-level generic instantiations, for instance. OTOH, I was under the impression that most Ravenscar programs used what is now known as the ["Sequential"] elaboration policy (see H.6 in the Ada 2005 RM). That prevents tasks from being started until elaboration of library units is finished. So the problem is less that you have to do things at elaboration time in Ravenscar, and more that Ravenscar is incompatible with the traditional Ada elaboration model.

From: Matthew Heaney <matthewjheaney@earthlink.net>
Subject: Re: Elaboration worries
Date: Thu, 22 Jun 2006 02:24:24 GMT
Newsgroups: comp.lang.ada

[…] you need to pragma Elaborate_All whenever you do an instantiation, e.g.

```
with GP;
pragma Elaborate_All (GP);

package Q is
   pragma Preelaborate;
   -- or whatever

   package P is new GP (…);
   …
end Q;
with GR;
pragma Elaborate_All (GR);
package body Q is
  package R is new GR (…);
  …
end Q;
```

The Charles library and the GNAT implementation of the standard container library are implemented like that, with an Elaborate_All on the generic package being instantiated.

Note that I don't usually bother using pragma Elaborate_Body unless I need to either force a body for a spec that otherwise wouldn't require a body, or because the body has state. In the latter case you want to ensure that the package state is fully elaborated before any operations in that package are called (by some other package during its own elaboration).

Of course if, during elaboration of a package, the package calls an operation in some other package, then the package must Elaborate_All on the called package.

Normally you want pragma Elaborate_All, but pragma Elaborate is still useful occasionally, when elaborating packages with mutual dependencies.

From: Alex R. Mosteo <devnull@mailinator.com>
Subject: Re: Elaboration worries
Date: Thu, 22 Jun 2006 18:25:15 +0200
Newsgroups: comp.lang.ada

For the record, I've found this link a interesting read in relation with this topic:

http://www.ada-auth.org/cgi-bin/cvsweb.cgi/AIs/AI-00366.TXT?rev=1.19

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Subject: Re: Elaboration worries
Date: 21 Jun 2006 19:12:41 -0400
Newsgroups: comp.lang.ada

I have put debug output code in Pure and Preelab packages by "cheating". Like this: write a simple I/O package (I like to avoid the complexity of Text_IO). Use pragma Export(Ada) on all of its procedures. Then write another package that declares all the same procedures, with pragma Import(Ada), and put pragma Pure in that.

This is cheating, so I only do it for temporary debugging output.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Elaboration worries
Date: Thu, 22 Jun 2006 18:09:50 -0500
Newsgroups: comp.lang.ada

True, you can leave the language if you like, or lie to the compiler (I guess you'd say what you are doing is the latter; I would call it the former), but I was thinking of ways that don't require such underhandedness and can be used in production code.

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: Elaboration worries
Date: Thu, 22 Jun 2006 18:31:18 -0500
Newsgroups: comp.lang.ada

> I'm also interested in the "cheating" for I/O, since this seems really a weak spot. Any other cheats besides the one mentioned by Mr.Duff?

Well, I don't cheat, rather I pass in the logging routines to the Preelaborated packages. (This will also work for Pure in Ada 2005.)

First, I declare a logging access type:

```
type Logger_Access is access
   procedure (Message : in String);
   -- Write a line to the log (if any).
```

And then all of the routines that need to do logging including a Logger parameter:

```
Logger : in Logger_Access := null
```

If Logger is null, nothing is written. Logger_Access is defined so that the profile matches Ada.Text_IO.Put_Line for unit debugging, and it matches the profile of the logging routines that we usually use.

Now, you do have to pass this through all of the calls inside of Preelaborated packages. Once you get to a normal package, you can just provide the appropriate logger routine:

```
…
Logger =>
   Ada.Text_IO.Put_Line'Access);
```

And this provides a way to access "normal" stuff from Preelaborated packages. Of course, it only works if the number of such things is rather limited.

I should point out that I started using this because I had a need to use different logging techniques in different programs that depended on the same shared library. The fact that it let some of the libraries be Preelaborated was a bonus.

Debug and Conditional Compilation

From: Guillaume Portail <guillaume.portail@bigfoot.com>
Subject: Conditional compilation of debug traces without cpp
Date: Tue, 04 Jul 2006 20:06:51 +0200
Newsgroups: comp.lang.ada

I have pieces of code like this:

```
package Debug is
   procedure Put_Line (M : in String);
end;
package body Debug is
   Enabled : constant Boolean :=
      False;
   procedure Put_Line (M : in String)
      is
   begin
    if Enabled then
       Real_Put_Line (M);
    end if;
   end;
end;
```

Many other units, many calls like:

```
…
Debug.Put_Line ("PC was here, A =
  ", A_Type'Image(A));  -- (1)
…
```

Debug.Enabled is a compile time constant, so with a bit of -O3 pragma Inline or other (-gnatN), the binary implementation of Debug.Put_Line will be null. But never will be the elaboration of the many calls to it. I guess that the elaboration of theses calls is always required by the language, think of :

```
Debug.Put_Line ("PC was here, A = "
  & A_Function_Call(12));  -- (2)
Debug.Put_Line ("PC was here, A = "
  & A_Function_Call(1/0)); -- (3)
```

For my needs, A_Function_Call is only for debugging purposes here, it has no side effects (it is a function).

How may I organize the code to obtain the effect of having Debug.* calls being nulls when Debug.Enabled is False?

For your information, this is easy using cpp:

```
#if _DEBUG
#define DEBUG(x)
       real_put_line x
#else
#define DEBUG(x)    0
….
   if (foo)
   {
     a = something;
     DEBUG(("PC was here,
       A=%d", b()+a));
   }
   else
     DEBUG(("no foo"));
….
```

And I would like not to use cpp or gnatprep, etc.

From: Guillaume Portail <guillaume.portail@bigfoot.com>
Subject: Re: Conditional compilation of debug traces without cpp
Date: Tue, 04 Jul 2006 21:14:26 +0200
Newsgroups: comp.lang.ada

> You could place all calls to Debug.Put_Line inside a pragma Debug. These are enabled only if you pass -gnata to GNAT; they are off by default.

It helps, but it is GNAT specific. I would prefer a pure Ada solution.

From: Matthew Goulet <blueherring0@gmail.com>
Subject: Re: Conditional compilation of debug traces without cpp
Date: 4 Jul 2006 20:10:31 -0700
Newsgroups: comp.lang.ada

The reference manual specifies unknown pragmas should be ignored, so if development is done on GNAT pragma Debug could be used, and they'd be ignored on other compilers (sort of the desired effect). Not exactly compiler agnostic, but a thought.

From: Björn Persson <rombo.bjorn.persson@sverige.nu>
Subject: Re: Conditional compilation of debug traces without cpp
Date: Tue, 04 Jul 2006 21:39:33 GMT
Newsgroups: comp.lang.ada

Well, there is always the good old if statement:

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Debug_Demo is
   Debug_Enabled: constant
     Boolean := false;
begin
   if Debug_Enabled then
     Put_Line("PC was here, A = " &
       Integer'Image(12));
   end if;
end Debug_Demo;
```

When Debug_Enabled is false, the compiler (at least Gnat) will remove the if statement completely.

From: Guillaume Portail <guillaume.portail@bigfoot.com>
Subject: Re: Conditional compilation of debug traces without cpp
Date: Tue, 04 Jul 2006 23:43:34 +0200
Newsgroups: comp.lang.ada

This complicates the code (the McCabe numbers), it is better to have the if statement encapsulated.

From: Gautier <gautier@fakeaddress.nil>
Subject: Re: Conditional compilation of debug traces without cpp
Date: Tue, 04 Jul 2006 22:24:44 +0200
Newsgroups: comp.lang.ada

At least this works, with GNAT 3.15p and ObjectAda 7.2.2 SE:

```
package Debug is
  procedure Put_Line (M : in String);
  pragma Inline(Put_Line);
  procedure Put_Line (M : in String;
                      I: Integer);
  pragma Inline(Put_Line);
end Debug;
with Ada.Text_IO;
package body Debug is
  Enabled : constant Boolean :=
    False;
  procedure Put_Line (M : in String)
    is
  begin
   if Enabled then
     Ada.Text_IO.Put_Line (M);
   end if;
  end;
  procedure Put_Line (M : in String;
                      I: Integer) is
  begin
   if Enabled then
     Ada.Text_IO.Put_Line
       (M & Integer'Image(I));
   end if;
  end;
end Debug;
with Debug, Ada.Text_IO;
procedure Test_debug is
begin
  for I in 1..1234 loop
   Ada.Text_IO.Put_Line("[a]");
   Debug.Put_Line
     ("(0) PC was here");  -- (0)
   Debug.Put_Line
     ("(1) PC was here,
      I = ", I);  -- (1)
   Ada.Text_IO.Put_Line("[b]");
  end loop;
end;
```

gcc -O2 -S -gnatpN test_debug.adb

[…] When you have concatenations and/or a function call in the parameter it doesn't work with these compilers — maybe, as you guess, they can't skip the computation of parameters. Or they may, or newer versions do. Perhaps it is a question of having a pragma Pure for the function.

From: Simon Wright <simon@pushface.org>
Subject: Re: Conditional compilation of debug traces without cpp
Date: Tue, 04 Jul 2006 21:54:05 +0100
Newsgroups: comp.lang.ada

> Debug.Enabled is a compile time constant, so with a bit of -O3 pragma Inline or other (-gnatN), the binary implementation of Debug.Put_Line will be null.

I don't think you need more than -O2 for this magic to work. But it's a promise made by GNAT, other compilers may behave differently.

You could check out gnatprep, I suppose.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: Re: Conditional compilation of debug traces without cpp
Date: Wed, 05 Jul 2006 15:03:42 +0200
Organization: Adalog
Newsgroups: comp.lang.ada

Small plug: have a look at package Debug from Adalog's components page (http://www.adalog.fr/compo2.htm), it does this in a very sophisticated way…

As for your question, name you package Debug_Effective. Then write another package (Debug_Dummy) with the same specification, and where all procedures are null. Then simply compile:

```
with Debug_Effective;
package Debug
  renames Debug_Effective;
```

or:

```
with Debug_Dummy;
package Debug
  renames Debug_Dummy;
```

Switching packages is then not more difficult than changing a #define…

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: Re: Conditional compilation of debug traces without cpp
Date: Thu, 06 Jul 2006 09:59:55 +0200
Organization: Adalog
Newsgroups: comp.lang.ada

Forgot to mention: make the fake debug pure, to avoid any elaboration.[…]

Therefore yes, the *execution* of the call will always involve the *evaluation* of the parameters (unless the compiler can prove that this evaluation has no side-effects — like a call to a function declared in a pure package).

Ripple effect

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Subject: Re: Ripple effect
Date: 04 Sep 2006 09:52:01 -0400
Newsgroups: comp.lang.ada

[…]

> I recall that during the Ada-9X revision process, it was proposed that primitive operators of a type have this kind of visibility. IIRC, one of the reasons that this was not accepted was that it would lead to Ripple effects: adding or removing a unit from a context clause could change one legal program to a different legal program.

No, I think you misunderstand the "Ripple Effect". As I understand it, the Ripple Effect means that adding/removing a with_clause can cause compilation units that do not depend DIRECTLY on the modified thing to become illegal. For example, suppose C with's B and B with's A. Can a with_clause on A affect the legality of C? If so, there's a Ripple Effect.

There's no issue of changing the meaning from one legal program to another, as can happen with the Beaujolais Effect. Therefore, the Ripple Effect is (IMHO) merely an annoyance, rather than a bug-causing language-design flaw.

I believe Tucker coined the term "Ripple Effect", and that one is supposed to imagine the addition of a with_clause rippling through the transitive closure of semantic dependences. Also, it's a joke because Ripple is a wine — that's the main similarity to Beaujolais Effect. I've never tasted Ripple, but I think it's considered to be of somewhat lower quality than Beaujolais. ;-)

Anyway, to answer Jeff's question: I think with_clauses should be transitive in the first place, so that the Ripple Effect is not an issue.

One problem with Ada 83 is that with_clauses cannot appear inside the private part. I think that's the root of the idea that transitive with_clauses are somehow evil.

[See also "The Beaujolais Effect Revisited" in AUJ 23-1 (Mar 2002), pp.38–39. —su]

From: Jeffrey R. Carter <jrcarter@acm.org>
Subject: Re: Ripple effect
Date: Mon, 04 Sep 2006 15:15:05 GMT
Newsgroups: comp.lang.ada

'In brief, the (undesirable) Ripple effect was related to whether the legality of a compilation unit could be affected by adding or removing an otherwise unneeded "with" clause on some compilation unit on which the unit depended, directly or indirectly.' (Tucker Taft)

So it's not a with on C, as I thought, and can be a with on B, which you exclude. It also refers only to unneeded withs. So, if B withs A unnecessarily, that could cause a Ripple effect.

Teaching a Specification Language

From: Colin Paul Gloster <Colin_Paul_Gloster@ACM.org>
Subject: Re: Generating Ada from UML on Linux
Date: Wed, 9 Aug 2006 13:01:18 +0200
Newsgroups: comp.lang.ada

> I have to teach a course on software design with UML [..]

Perhaps you should refuse to teach an inadequate modeling language which is not a formal specification language.

From: Jacob Sparre Andersen <sparre@nbi.dk>
Subject: Re: Generating Ada from UML on Linux
Date: Thu, 10 Aug 2006 12:59:44 +0200
Newsgroups: comp.lang.ada

Perhaps I should, but initially I am just refusing to teach the students Java. And they will of course also be explained the limitations of UML.

Which formal specification language would you suggest for teaching undergraduate students? I am open to suggestions. Formal specifications is not exactly my primary expertice. I can easily see that UML is much too weak, compared to what we can implement in Ada, but I am not aware of any more practical choices.

From: Colin Paul Gloster <Colin_Paul_Gloster@ACM.org>
Subject: Re: Generating Ada from UML on Linux
Date: Thu, 10 Aug 2006 19:06:54 +0200
Newsgroups: comp.lang.ada

I am not sure which to suggest, a number of them also have weaknesses and I do not know all of the good specification languages. VDM (specifically VDM-SL) can be represented in ASCII but it has no reuse capability […] Z has this basic capability but is not expressible in ASCII (some people write it in LaTeX but LaTeX can easily need several runs before all interdependant references are resolved but still produces output which if not thoroughly checked could be mistaken for a finished document). Also, Z does not have any built-in temporal facilities.

Being an Ada course, the SPARK specification language might not be irrelevant.

From: Stephen Leake <stephen_leake@acm.org>
Date: Sat, 12 Aug 2006 08:07:05 -0400
Subject: Re: Generating Ada from UML on Linux
Newsgroups: comp.lang.ada

In no sense is UML a "formal specification language"!

Ada is a better specification language than UML, if you are looking for well-defined execution semantics.

As far as I can tell, the only advantage UML has over Ada is that you can write it with a mouse — and personally, I consider that a disadvantage :).

I understand there are good tools for Z (much better than just LaTeX), but I have not used them.

From: Jacob Sparre Andersen sparre@nbi.dk
Subject: RSL (Was: Generating Ada from UML on Linux)
Date: Tue, 15 Aug 2006 16:27:37 +0200
Newsgroups: comp.lang.ada

I took a look at the table of contents for Dines Bjørner's "Software Engineering: Abstraction and Modelling". It looked much too theoretical for my students. I have no intention of teaching them the complete theoretical foundations of RSL, which seems to be the aim with Dines Bjørner's book. Are there some more practically oriented introductions to RSL?

From: Mark Lorenzen <mark.lorenzen@surfpost.dk>
Date: 16 Aug 2006 02:10:44 +0100
Subject: Re: RSL (Was: Generating Ada from UML on Linux)
Newsgroups: comp.lang.ada

[…] Dines usually focuses on practical applications when writing text books. There are "pure" RSL books available, but they are definitely more theoretical, although still accessible.

There is of course always the problem of tool support (or lack of) when one wants to use RSL.

Bug Reports

From: M E Leypold <kontakt@m-e-leypold.de>
Date: 21 Jun 2006 14:29:59 +0200
Subject: Re: Compiler Bug or what I'm doing wrong?
Newsgroups: comp.lang.ada

> It would be nice if you could write a minimal test case that reproduces the problem. Yes, it takes time. I cannot do that for you, for various

I've already done that (it took me 10 hours so far to isolate the problem from a larger program): The original program was much larger. I'll be trying to even strip the example further, but the problem with the bug seems to be that it's a Heisenbug which vanishes when you delete some fields in the datastructure, then turns up again if you delete more fields and so on. The malloc() implementation of libc which the GNAT runtime uses is unchecked so obviously one gets sometimes away when freeing invalid pointers and (as we know from C programming).

Whatever: I'll try to produce an even smaller test case.

From: James Dennett <jdennett@acm.org>
Subject: Re: Compiler Bug or what I'm doing wrong?
Date: Wed, 21 Jun 2006 19:07:31 -0700
Newsgroups: comp.lang.ada

> Indeed, for me the most frustrating part of bug reporting is obtaining a sufficiently small test case. I usually hit bugs in a large project I'm involved. Once you start to chop off, you know that the bug will disappear along the way. When you're not even sure if you're the culprit and not the compiler, the frustration is even bigger.

The "delta" tool can largely automate the task of reducing a test case to something closer to minimal, though some manual intervention helps:

http://gcc.gnu.org/wiki/A_guide_to_testcase_reduction

I've used it only a couple of times, but it saved me a *lot* of effort. (On the other hand, it might need tweaking to support Ada; I don't know, I used it on C++ code.)

[See also http://delta.tigris.org/ for more information about this tool --su]

From: Simon Wright <simon@pushface.org>
Subject: Re: Compiler Bug or what I'm doing wrong?
Date: Thu, 22 Jun 2006 20:10:40 +0100
Newsgroups: comp.lang.ada

> It's a pity that GNAT (at least 3.15p? I'm repeating myself …), doesn't have some kind of community support side where things as these could be collected. Or is there such a site?

What about the GNU Ada wiki at Sourceforge?
http://gnuada.sourceforge.net/

From: Simon Wright <simon@pushface.org>
Subject: Re: Compiler Bug or what I'm doing wrong?
Date: Sat, 24 Jun 2006 21:33:10 +0100
Newsgroups: comp.lang.ada

> After inspecting more carefully the gnuada site, I believe that the bug tracker is for their installation packages. Maybe the 3rd party bugs section could serve.

I would have thought something starting at the GNATP page (== 3.15p) would be appropriate for that compiler, similar for others. This is at http://gnuada.sourceforge.net/pmwiki.php/Packages/GNATP reached from the Packages section of the first page.

From: M E Leypold <kontakt@m-e-leypold.de>
Date: 26 Jun 2006 13:16:24 +0200
Subject: Re: Compiler Bug or what I'm doing wrong?
Newsgroups: comp.lang.ada

> I would suggest the Debian bug tracker instead; not because I'm trying to promote Debian, but because the Debian BTS already has 200 bugs filed in it or so.

I'll try to file the bug in both places. IMHO gnuada.sourceforge.net would be the logical place for tracking bugs in community supported Ada software, but since nobody has reported anything there yet, one can hardly expect anyone looking there for solutions.

> The community will be better served by one central bug database than by several disjoint ones.

I completely agree. I think the situation is a bit confused presently because of the transition between p-releases and GCC GNAT: For GCC gnat the upstream source for the Debian releases would come from http://gcc.gnu.org/ and their BTS would be the first to report bugs to. Debian BTS would either mostly refer to bugs in the GCC BTS or to packaging defects.

[…] I think the situation wrt to where to report bugs will become more clearly defined when the transition to gccada has been done.

> I specifically encourage anyone to submit bugs there, even if they do not use Debian. Similarly, my patches that fix several bugs in 3.15p are available to all.

Yes, I already perceived that Debian is the only open platform where I'd say that Gnat is actually supported in sense.

> The only reason why I started using the Debian BTS back in 2003 was because no other public database existed at that time. So, I claim precedence :)

You'll get it.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: [Ada in Debian] GtkAda and GNAT versions
Date: 26 Jun 2006 05:13:58 -0700
Newsgroups: comp.lang.ada

The Debian documentation says you should report bugs to the Debian BTS, and nowhere else. The package maintainer then decides whether the bug comes from the packaging, or from upstream. In the case of gnat 3.15p, there simply is no upstream bug database. In the case of gnat-4.1 (the default in Etch), we use GCC's bugzilla as the upstream database.

[…] People not using Debian are welcome to reports bugs, but I'd ask them to state exactly what platform they're using, as with any bug report, but also to check that their bug is not already solved in Debian.

Calling a System Command

From: Jim Maureen Rogers <jimmaureenrogers@worldnet.att.net>
Subject: Re: Is there some way of calling a system command?
Date: 11 Jun 2006 06:19:12 -0700
Newsgroups: comp.lang.ada

> I am a newbie Ada-programmer and need help with this topic. In C one can launch a system command, for instance through the following function.
> `system("ls ");`
> or in Java
> `exec("ls");`
> Is there some similar way of calling a system command in Ada?

The easiest way is to simply call the C system command. The example below was run on my Windows XP system.

```
with Interfaces.C.Strings;
use Interfaces.C.Strings;
procedure System_Example is
  function System_Call
    (Command : Chars_Ptr) return
      Interfaces.C.Int;
  pragma Import
    (Convention => C,
     Entity => System_Call,
     External_Name => "system");
  Rc : Interfaces.C.Int;
begin
  Rc :=
    System_Call(New_String("dir"));
end System_Example;
```

The important part of the example is the creation of a function specification I have named System_Call. That function specification takes a parameter of Chars_Ptr, which corresponds to a C char

*. The function specification is used as the Ada interface to the C system call. The compiler is notified of that association through the pragma Import. That pragma causes the compiler to link the C system command and call it whenever System_Call is called in the Ada code.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: Re: Is there some way of calling a system command?
Date: 15 Jun 2006 07:11:10 -0700
Newsgroups: comp.lang.ada

> I am learning Ada and do not know C. I tried your approach to call an other program from Ada and it worked, except for one problem: the calling program will wait for the return parameter (Rc in your example) and thus it will freeze until the other one is closed. Could I do something similar but withour the return parameter?

See GNAT.OS_Lib.Non_Blocking_Spawn, GNAT.Expect.Non_Blocking_Spawn.

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.

2006

October 01-06 — 9th **International Conference on Model-Driven Engineering Languages and Systems** (MoDELS'2006), Genoa, Italy. Topics include: Model-driven engineering methodologies, approaches, languages and tools; Domain-specific modeling languages; Programming language and meta-programming support for linking models to code; Modeling languages and tools; Semantics of modeling languages; Modeling and analysis of real-time, embedded, and distributed systems; etc.

☺ October 02-04 — 25th **IEEE International Symposium on Reliable Distributed Systems** (SRDS'2006), Leeds, UK. Topics include: reliability, availability, safety, security, and real time; Security and high-confidence systems, Distributed objects and middleware systems, Formal methods and foundations for dependable distributed computing, Analytical or experimental evaluations of dependable distributed systems, etc.

October 02-06 — 20th **Brazilian Symposium on Software Engineering** (SBES'2006), Florianópolis – Brazil. Topics include: Object-oriented Development; Component-based Software Engineering; Distributed Software Engineering; Empirical Software Engineering and Metrics; Model Driven Development; Multi-paradigm and Multi-language Modelling and Programming; Object-oriented Techniques; Software Economics; Software Engineering for Embedded and Real-time Software; Software Engineering Tools and Environments; Software Maintenance and Reverse Engineering; Software Quality; Software Reuse; Software Safety, Dependability, and Reliability; Software Security; Software Verification, Validation and Inspection; etc

October 11-13 — **European Systems and Software Process Improvement and Innovation Conference** (EuroSPI'2006), Joensuu, Finland.

☺ October 12-13 — **Automotive - Safety & Security 2006**, Stuttgart, Germany. Theme: "Sicherheit und Zuverlässigkeit für automobile Informationstechnik". Organized by Gesellschaft für Informatik (GI), etc., in cooperation with Ada-Deutschland and Fachgruppe "Ada", etc. Includes keynote by Prof. Michael Gonzales (Univ. de Cantabria, Spain) on "Predictable Response Times in Event-driven Real-time Systems". Topics include (in German): Zuverlässigkeit und Sicherheit für fahrbetriebs-kritische Software und IT-Systeme; Sichere Entwicklung, Aktualisierung und Freischaltung; Normen und Standardisierungsbestrebungen; Entwicklungsbegleitende Evaluation und Zertifizierung; etc.

October 16-20 — 10th **International IEEE Enterprise Distributed Object Computing Conference** (EDOC'2006), Hong Kong.

☺ October 18-20 — **IEEE Symposium on Industrial Embedded Systems** (IES'2006), Antibes, Juan les Pins, Cote d'Azur, France. Topics include: recent developments, deployments, technology trends and research results, as well as initiatives related to embedded systems and their applications in a variety of industrial environments.

☺ October 22 — **Workshop on Linguistic Support for Modern Operating Systems** (PLOS'2006), San Jose, California, USA. Topics include: type-safe languages for OS; language-based security and OSs; language support for OS verification, testing, and debugging; etc.

☺ October 22-26 — 21st **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2006), Portland, Oregon, USA. Topics include: diverse disciplines related to object technology.

October 22-26 — 5th **International Conference on Generative Programming and Component Engineering** (GPCE'2006), Portland, Oregon, USA. Co-located with OOPSLA'2006. Topics include: Generative

techniques for Product-line architectures; Distributed, real-time and embedded systems; Model-driven development and architecture; Component-based software engineering (Reuse, distributed platforms and middleware, distributed systems, evolution, patterns, development methods, deployment and configuration techniques, and formal methods); Integration of generative and component-based approaches; Industrial applications; etc.

October 23 — 2nd **International Workshop on Code Based Software Security Assessments** (CoBaSSA'2006), Benevento, Italy. Topics include: Mitigating stack- or heap-based buffer overflow attacks; Race condition detection; Case studies in analyzing software vulnerability; Best practices for secure coding; etc. Deadline for early registration: October 7, 2006.

October 23-26 — 4th **International Symposium on Automated Technology for Verification and Analysis** (ATVA'2006), Beijing, China. Topics include: theory useful for providing designers with automated support for obtaining correct software or hardware systems, applications of theory in engineering methods and particular domains and handling of practical problems occurring in tools, etc.

October 23-27 — 13th **Working Conference on Reverse Engineering** (WCRE'2006), Benevento, Italy. Theme: "Empirically Assessing Reverse Engineering Techniques and Tools". Topics include: Empirical studies in reverse engineering; De-compilation and binary translation; Re-documenting legacy systems; Reverse engineering tool support; Mining software repositories; Program analysis and slicing; Software architecture recovery; Program transformation and re-factoring; etc.

October 25-27 — 5th **International Conference on Software Methodologies Tools, and Techniques** (SoMeT'2006), Quebec, Canada. Topics include: Software methodologies, and tools for robust, reliable, non-fragile software design; Automatic software generation versus reuse, and legacy systems, source code analysis and manipulation; Software evolution techniques; Formal methods for software design; Static and dynamic analysis, and software maintenance; Formal techniques for software representation, software testing and validation; Software reliability, and software diagnosis systems; etc.

October 26-28 — 6th **International Conference on Quality Software** (QSIC'2006), Beijing, China. Topics include: Software quality (reliability, safety and security, ...); Methods and tools; Evaluation of software products and components (static and dynamic analysis, validation and verification); Formal methods (program analysis, model checking, formal process models, ...); Applications (component-based systems, distributed systems, embedded systems, enterprise applications, safety critical systems, ...); etc.

☺ Oct. 29-Nov. 03 — 8th **International Symposium on Distributed Objects and Applications** (DOA'2006), Montpellier, France. Topics include: Application case studies of distribution technologies; Component-based software development; Design patterns for distributed systems; Integrated development environments; Middleware for distributed object computing; Real-time solutions for distributed objects; Technologies for reliability and fault-tolerance; Testing and validation of distributed object systems; etc.

Oct. 30-Nov. 03 — 8th **International Conference on Formal Engineering Methods** (ICFEM'2006), Macao SAR, China. Topics include: Abstraction and refinement; Tool development and integration for formal system design, analysis and verification; Integration of formal verification tools in CASE tools; Techniques for specification, verification and validation; Techniques and case studies for correctness by construction; Experiments of verified systems; Application in real-time, hybrid and critical systems; Emerging technologies; etc.

November 01-03 — 21st **International Symposium on Computer and Information Sciences** (ISCIS'2006), Istanbul, Turkey. Topics include: Computer Architecture and Embedded Systems, Parallel and Distributed Computing, Security & Cryptography, Software Engineering, Theoretical Computer Science, etc.

November 08-10 — 4th **Asian Symposium on Programming Languages and Systems** (APLAS'2006), Sydney, Australia. Topics include: both foundational and practical issues in programming languages and systems; type systems, language design; program analysis, optimization; software security, safety, verification; compiler systems, interpreters; programming tools and environments; etc.

♦ Nov. 12-16 — 2006 **ACM SIGAda Annual International Conference** (SIGAda'2006), Albuquerque, New Mexico, USA. Sponsored by ACM SIGAda, in cooperation with SIGAPP, SIGCAS, SIGCSE, SIGPLAN, SIGSOFT, Ada-Europe, and Ada Resource Association (ACM approval pending, Cooperation approvals pending.). Topics include: reliability needs and styles; safety and high integrity issues; analysis, testing, and

validation; standards; use of ASIS for new Ada tool development; mixed-language development; Ada in XML and .NET environments; quality assurance; Ada & software engineering education; commercial Ada applications: what Ada means to the bottom line; static and dynamic code analysis; software architecture and design; etc. Keynote presentations: Judith Klein, Lockheed Martin, "Use of Ada in Lockheed Martin for Air Traffic Management and Beyond"; Robert Dewar, AdaCore, "Ada 2005 & High Integrity Systems"; Tucker Taft, SofCheck, "Why You Should be Using Ada 2005 now!" Deadline for early registration: October 15, 2006.

November 20-24 3rd **International Colloquium on Theoretical Aspects of Computing** (ICTAC'2006), Gammarth/Tunis, Tunisia. Topics include: principles and semantics of programming languages; software architectures and their description languages; software specification, refinement, and verification; model checking and theorem proving; models of object and component systems; integration of formal and engineering methods; models of concurrency; theory of parallel and distributed computing; real-time and embedded systems; etc.

☺ Nov. 30-Dec. 01 **National Workshop on High Confidence Software Platforms for Cyber-Physical Systems: Research Needs and Roadmap**, Alexandria, Virginia, USA. Topics include: producing distributed, real-time, and embedded platforms and applications, where computer processors control physical, chemical, or biological processes or devices; R&D strategies and tactics for restructuring the current real-time operating system, virtual machine, and distributed computing middleware platforms into a sound and assured real-time technology base for building future cyber-physical systems. Deadline for submissions: October 18, 2006, 2006

☺ December 01-04 4th **International Symposium on Parallel and Distributed Processing and Applications** (ISPA'2006), Sorrento, Italy. Topics include: Parallel/distributed system architectures; Tools and environments for software development; Parallel/distributed algorithms; Distributed systems and applications; Reliability, fault-tolerance, and security; etc. Includes "Languages and Algorithms" and "Software and Applications" Tracks.

☺ December 04-07 7th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2006), Taipei, Taiwan. Topics include: Parallel/distributed architectures; Reliability, and fault-tolerance; Formal methods and programming languages; Parallelizing compilers; Component-based and OO Technology; Tools and environments for software development; Parallel/distributed algorithms; Task mapping and job scheduling; etc.

December 05-07 19th **International Conference on Software & Systems Engineering and their Applications** (ICSSEA'2006), Paris, France. Topics include: distributed systems, real-time systems, embedded systems, interoperability, evolution, object-orientation, formal methods, validation, certification, reliability, etc.

☺ December 05-08 27th **IEEE Real-Time Systems Symposium** (RTSS'2006), Rio de Janeiro, Brazil. Topics include: all aspects of real-time systems design, analysis, implementation, evaluation, and case-studies.

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

December 15 **BCS-FACS Christmas Meeting 2006: Teaching Formal Methods Workshop**, London, UK. Topics include: how to motivate the study of formal methods; linking formal methods and software development; tools for teaching formal methods; etc. Deadline for submissions: October 20, 2006.

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; ...); etc.

January 15-16 **ACM SIGPLAN 2007 Symposium on Partial Evaluation and Program Manipulation** (PEPM'2007), Nice, France. Co-located with POPL'2007. Topics include: program manipulation, partial evaluation, and program generation. PEPM focuses on techniques, theory, tools, and applications of

analysis and manipulation of programs. Deadline for submissions: October 18, 2006 (abstracts), October 20, 2006 (papers).

☺ 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 **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. Deadline for submissions: October 6, 2006

☺ February 07-09 15th **Euromicro Conference on Parallel, Distributed and Network-based Processing** (PDP'2006), Naples, Italy. Topics include: Advanced Applications (scientific and engineering applications, multi-disciplinary 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.

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

March 11-15 22nd **ACM Symposium on Applied Computing** (SAC'2007), Seoul, Korea.

- ☺ 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.

* Mar. 24-Apr. 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. Deadline for submissions: October 6, 2006 (abstracts), October 13, 2006 (papers).

☺ 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. Deadline for submissions: October 9, 2006.

☺ Mar. 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. Deadline for submissions: November 6, 2006.

☺ Mar. 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. Deadline for submissions: November 6, 2006.

March 27-29 13th **Conference on Languages and Models with Objects** (LMO'2007), Toulouse, France. Deadline for submissions: October 13, 2006 (abstracts), October 20, 2006 (papers).

♦ 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. Deadline for submissions: October 13, 2006.

☺ May 20-26 29th **International Conference on Software Engineering** (ICSE'2007), Minneapolis, Minnesota, USA. Deadline for submissions: October 8, 2006 (education papers, tutorials, workshops), October 29, 2006 (research demonstrations), December 11, 2006 (doctoral symposium).

☺ May 29-June 01 **DAta Systems In Aerospace** (DASIA'2007), Naples, Italy.

June 18-21 **Systems and Software Technology Conference** (SSTC'2007), Tampa (Florida, USA).

June 25-27 12th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2007), Dundee, Scotland, UK.

☺ June 09-16 3rd **History of Programming Languages Conference** (HOPL-III), San Diego, CA, USA. Co-located with FCRC'2007.

♦ June 25-29 12th **International Conference on Reliable Software Technologies - Ada-Europe'2007**, Geneva, Switzerland. Sponsored by Ada-Europe, in cooperation with ACM SIGAda (approval pending). Deadline for submissions: November 6, 2006 (papers, tutorials, workshops).

June 25-29 27th **International Conference on Distributed Computing Systems** (ICDCS'2007), Toronto, Canada. Topics include: all aspects of distributed and parallel computing. Deadline for submissions: November 20, 2006 (papers).

☺ 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 technology. Deadline for submissions: February 1, 2007 (draft papers).

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

☺ 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.

13TH INTERNATIONAL REAL-TIME ADA WORKSHOP
IRTAW-13

17-19 April 2007
Woodstock, Vermont
USA

CALL FOR PARTICIPATION

For over 20 years the series of **International Real-Time Ada Workshop** meetings has provided a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions, and has attracted participation from key members of the research, user, and implementer communities worldwide. Since the standardization of Ada 95, the **IRTAW** series has assisted in the review of the real-time related portions of the *Guide for the Use of the Ada Programming Language for High Integrity Systems* (ISO/IEC TR 15942:2000) and has developed and promoted the Ravenscar tasking profile. Recent IRTAW meetings have significantly contributed to the new Ada 2005 standard, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar profile.

In keeping with this tradition, and in light of the formal approval of the Ada 2005 language standard, the goals of **IRTAW-13** will be to:

- examine early experiences in using Ada 2005 for the development of real-time systems and applications;
- report on or illustrate implementation approaches for the new real-time features of Ada 2005;
- consider the added value of developing other real-time Ada profiles in addition to the Ravenscar profile;
- examine the implications to Ada of the growing use of multiprocessors in the development of real-time systems, particularly with regard to predictability, robustness, and other issues;
- examine and develop paradigms for using Ada 2005 for real-time distributed systems, taking into account robustness as well as hard, flexible and application-defined scheduling;
- consider the definition of specific patterns and libraries for real-time systems development in Ada;
- identify how Ada relates to the certification of safety-critical and/or security-critical real-time systems;
- review the status and contents of the current ISO reports related to real-time Ada and consider the interest of developing new secondary standards or extensions;
- examine the status of the Real-Time Specification for Java and other languages for real-time systems development, and consider user experience with current implementations and with issues of interoperability with Ada in embedded real-time systems;
- consider the lessons learned from industrial experience with Ada and the Ravenscar Profile in actual real-time projects.

Participation at **IRTAW-13** is by invitation following the submission of a position paper addressing one or more of the above topics.

Position papers should not exceed six pages. All accepted papers will appear, in their final form, in the Workshop Proceedings, which will be published as a special issue of *Ada Letters* (ACM Press).

Please submit position papers, in pdf format, to the Program Chair by e-mail: **jpuente@dit.upm.es**

Program Committee

Ben Brosgol (Local Chair), Alan Burns, Michael Gonzalez Harbour, Stephen Michell, Javier Miranda, Luis Miguel Pinho, Juan Antonio de la Puente (Program Chair), Jorge Real, José Ruiz, Tullio Vardanega, Andy Wellings.

Important Dates

Receipt of Position Paper:	**12 January 2007**
Notification of Acceptance:	**16 February 2007**
Final Copy of Paper:	**16 March 2007**
Workshop Date:	**17-19 April 2007**

Memories of a Language Designer

Pascal Leroy

IBM Rational software; Chairman, Ada Rapporteur Group; email: pascal.leroy@fr.ibm.com

Introduction

On September 27, 2006, ISO/IEC JTC 1/SC 22, the ISO subcommittee in charge of standardizing programming languages, approved the Amendment to the Ada standard ("Ada 2005" in the vernacular) by twelve votes in favour[1], one abstention[2] and two non-voting countries supporting approval[3]. No comments were submitted as part of the vote, so the definition of Ada 2005 is now frozen as only administrative chores remain to be performed before the new standard is officially published by ISO. In parallel to this, the Ada standard will be published this fall in the prestigious Lecture Notes in Computer Science. After working on this Amendment for many years, it is now an interesting time to look back into the mirror.

1 Inception

Shortly after Ada 95 was standardized minor issues and questions concerning the language were addressed to the Ada Rapporteur Group (ARG). Though none of these were earth-shattering, there were anomalies and inaccuracies in the Reference Manual that needed fixing. In 1996 to address those issues the ARG started working on a Technical Corrigendum, which was completed in 2001.

In the meanwhile it became clear that there would not be financial room for another massive revision effort like the Ada 9X project. Future evolutions of the language would thus have to happen as part of a volunteer effort from people and organizations having an interest in Ada. As the user community started to use Ada 95 on real-life projects, they encountered a number of annoyances that could not be fixed by incremental changes. These early annoyances included for instance the impossibility of creating cyclic dependencies among package specifications, and the lack of support for interfacing with C or C++ unions.

Consensus quickly arose that it was necessary to put in place a framework for keeping Ada "alive" through a controlled revision process capable of preserving the benefits of standardization while also allowing room for improvements to the language.

2 History

As a consequence of those considerations, in 1998 the ARG was tasked by WG 9 (the ISO working group in charge of maintaining the Ada standard) to start studying "language enhancements". It is interesting to notice that by that time some of the most significant issues related to object-oriented programming and program structure were already identified: in addition to solutions to the cyclic dependencies problem, the suggestions made in 1998 included explicit control of overriding, upward-closure for subprogram parameters, and Java-style interfaces.

In fact, actual work on the Amendment did not really start until late 2000, in part because the focus was first on completing the Technical Corrigendum, and in part because more return on experience was needed before deciding what areas of the language actually required improvement.

In the summer of 2001 WG 9 asked for an Amendment to be developed with a target date of 2005 (this being software-related, it should not come as a surprise that we are now running some 9 months behind schedule on a 5-year project). WG 9 later approved a more formal and detailed schedule, as well as directions regarding the kinds of enhancements that the ARG should consider.

By that time it had become apparent that enhancements had to be developed to better support real-time and high-integrity systems, with the Ravenscar profile being the first item on the list. In the following years, numerous proposals relevant to this specific application area were developed by the International Real-Time Ada Workshop (IRTAW) and forwarded to the ARG for inclusion in Ada 2005 (though not without extensive rework in some cases!).

It had been clear from the beginning that expanding the predefined library was an essential goal of Ada 2005. Early on we worked on a package for accessing directories and file systems in a portable way, and we chose to include the matrix and vector facilities described in standard ISO/IEC 13813. Still, the topic that everyone had in mind was a predefined library of containers, though that looked like a daunting effort. To make that happen we decided to harness the help of the user community, and asked for proposals on this subject. After careful study of the two proposals that we received, we felt that neither of them was ideal. We thus decided to craft a third alternative by picking the good ideas in both submissions. This third alternative was initially given a very restricted scope, for we didn't want to miss the Amendment deadline. As it turned out, however once the core ideas had stabilized it was possible to add more packages, so that the final library may be deemed reasonably complete.

Interestingly the work on containers proved to be a valuable usability test for the new language: as difficulties were encountered in the development of the library, new features had to be added to the core language. Nested type

[1] Canada, China, Denmark, France, Germany, Japan, Romania, Russian Federation, Spain, Switzerland, United Kingdom, United States.

[2] The Netherlands.

[3] Belgium, Italy.

extensions and partial parameter parts for formal packages originated in this manner.

By the summer of 2004 the scope of the Amendment was pretty much stabilized, and it was clear which of the major proposals were in and which were out. What remained to be done was "mere" integration work. In fact, this proved much more time consuming than we anticipated. This had to do in part with the sheer size of the updated Reference Manual (nearly 1100 pages in its annotated version) and in part with the fact that upon reviewing all the changes "in place" we discovered inconsistencies that required rather extensive rework. For instance, some of the rules related to the inheritance of limited-ness, or to functions returning access results, came very late in the integration process and therefore required considerable attention. Overall, it took very intense work for everyone in the ARG over a period of 18 elapsed months before we had a document that could be submitted to WG 9 for approval.

3 Lessons Learned

It should not be surprising that "real" things take longer to finalize than one may initially expect. Some new features, like the limited_with_clause, proved extremely difficult to design: as many as seven proposals were considered over the course of five years before a satisfactory solution was arrived at: every time we tried a new idea, it seemed like it was breaking a fundamental invariant of the language. Interfaces have had a different story, but an equally complicated one at that: the basic ideas were essentially in place by the end of 2000, yet their entanglement with the rest of the language is so deep that we have been revising them literally until the last minute. Even apparently simple enhancements, like allowing aggregates for limited types, turned out to have unexpected consequences that required heart-wrenching decisions (for example, giving up on aggregates for private types).

It is amusing to notice that the ARG started out fairly timorous in the changes it contemplated: an early proposal for explicit control of overriding entailed making use of pragmas because the notion of adding new syntax was considered heretic. As it became clear that pragmas would be terrible for readability, we slowly warmed to the notion of new syntax. Thus we became increasingly bolder: when nested type extensions were added in 2003, they were easily swallowed, even though we knew that they would have a considerable impact on compilers.

While we were very much driven by the user community, a number of changes appeared out of our own work as we ran either into inconsistencies in Ada 95 or into unpleasant non-uniformities in Ada 2005. For instance, we initially added null_exclusions and improved support for anonymous access types so as to ease programming with access types (especially in the context of OOP). But for long we didn't want to allow anonymous access types as function results because of the deep language design difficulties that go with them. It was only when we started to use the new language for predefined units and for realistic examples that we discovered that it was an unacceptable limitation and decided to bite the bullet.

4 Regrets?

There are a number of ideas that we discussed for a long time, and on which considerable effort was expended, but which were not included in the final Amendment because we could not find a satisfactory solution. Partial generic instantiations come to mind, as do support for the IEEE 559 floating-point model, and pre- and post-conditions for types and packages. In all cases, integrating the new features into the language was hard and the solutions just didn't "feel right". Some of those ideas might in time mature to become valuable additions for a future version of Ada.

I suppose that every member of the ARG has his or her set of favourite features that didn't make it into the Amendment for one reason of another. Since I have the opportunity to do it here, I shall name my top three. I think it was unfortunate that we could not find a solution to the problem of partial generic instantiation, which hampers the usability of generics to compose abstractions; I guess that the discussion on this problem started too late and that we didn't have enough time to find the "magic" idea that would solve it elegantly. I also regret that we didn't have time to revise the exception mechanism: exceptions in Ada are frustratingly limited compared to other languages, but improving them without compromising performance and compatibility is a tough call. Finally, I wish we had had the guts to add **out** and **in out** parameters to functions, although I realize that this is a very controversial topic.

5 With (More Than) a Little Help…

It is often claimed that Ada was designed by a committee. Nothing could be further from the truth. The ARG is made of experts from the user community, the tools vendor community, and academia, who are among the best minds in our industry. It is important to stress that political bickering has no place in the ARG at all, and that although we have had a fair share of heated discussions, proposals were always judged on their technical merits. All the changes that were ultimately incorporated in the Amendment were elaborated by combining the best ideas, and were agreed upon quasi unanimously by the ARG.

I want to conclude by profusely thanking my fellow ARG members[4], who have devoted so much time and energy to bringing this effort to fruition, and the Convener of WG 9[5], who so deftly shepherded the Amendment through the Byzantine ISO administration. I have been lucky and honoured to work closely with all of them during all these years.

[4] Steve Baird, John Barnes, Randy Brukardt, Alan Burns, Robert Dewar, Gary Dismukes, Robert Duff, Kiyoshi Ishihata, Steve Michell, Erhard Ploedereder, Jean-Pierre Rosen, Ed Schonberg, Tucker Taft, Bill Thomas, Joyce Tokar, and Tullio Vardanega.

[5] James Moore.

Developing Reliable Software Rapidly

David N Kleidermacher

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

Abstract

Although there is significant evidence that following a structured, comprehensive quality management process improves reliability of software relative to the use of unstructured processes, these rigid methodologies often cause a loss in efficiency, delayed time to market, and frustration in the daily lives of software developers and managers. This paper will provide advice in the form of guidance statements that are compatible with the requirements of various high assurance quality standards, yet are designed and proven to improve efficiency of software development.

1 Introduction

Much commonality can be found in various quality management (e.g. ISO9000 and CMMI) and safety critical standards (e.g. IEC-61508 and RTCA/DO-178B). These standards promote a rigorous development process, covering such activities as configuration management, requirements specification, testing, quality policy enforcement, and quality maintenance. The traditional application of these standards, although often yielding higher quality and/or safer end products, often leads to stifling bureaucracy that slows time to market and innovation.

Proven in use since the early 1980s on reliability-critical systems such as automotive drive trains, aircraft engines, telecom switches, industrial plant controls, and medical devices, Green Hills Software's High Integrity Process is designed to maximize the reliability to production cost ratio of software. The process covers overreaching philosophies such as software system partitioning and change management as well as specific implementation details, such as the use of automated controls in the areas of testing, configuration management, and coding standard enforcement. The process has been applied successfully to meet the demands of ISO9000 audits, U.S. FDA class III (life critical) medical device maker audits, certification to RTCA/DO-178B, and certification to IEC-61508.

The Green Hills process can be easily adopted by software development teams that are currently suffering from the lack of a software quality management system and can be applied to legacy software projects written in common high level languages such as C. This paper will state a set of specific guidance rules of this process that enable highly reliable software to be developed while improving developer productivity and reducing time to market.

2 Partition Management

Many of the problems relating to loss in quality and safety in software can be attributed to the growth of complexity that can not be effectively managed [1]. An obvious solution to this problem is to decompose a large software system into smaller modules, each of which can be more easily understood and maintained. One of the key reasons why overly complex software is difficult to manage is that such a piece of software is almost always worked on by multiple developers, often at different times over the life of the product. Because the software is too complex for a single person to comprehend, features and defect resolutions alike are addressed by guesswork and patchwork. Flaws are often left uncorrected, and new flaws are added while attempting to correct other problems.

GUIDANCE #1: *ensure that no single partition is larger than a single developer can fully comprehend.*

Dividing a system into partitions requires that each partition have well-defined interfaces. Instead of hacking the same, shared piece of code, developers must define simple, clear interfaces for partitions and only use a partition's well documented (or at least well understood) interface to communicate with other partitions. Partitioning enables developers to work more independently and therefore more efficiently, minimizing time spent on meetings in which developers attempt to explain behavior of their software. Re-factoring a large software project in this manner can be time consuming. However, once this is accomplished, all future development will be more easily managed.

2.1 Enforcing Partitioning at Run Time

Usually, the embodiment of a partition in the target computer system is a single executable program. Examples of partitions include Windows .exe applications and UNIX processes. Thus, complex software made up of multiple partitions should always be used in conjunction with an operating system that employs memory protection to prevent corruption of one partition's memory space by another partition. Inter-partition communication is typically accomplished with standard message passing constructs, such as sockets or CORBA. Unless absolutely essential for better performance, shared memory should be avoided, since it blurs the lines of the designed separation.

Each partition must have a well-known *partition manager*. One way to ensure that developers understand who owns which partitions is to maintain an easily accessible partition manager list that is only modified by appropriate management personnel. The partition manager is the only person authorized to make modifications to the partition or to give another developer the right to make a modification. By having clear ownership of every single line of code in the project, developers are not tempted to edit code that they are not appropriately qualified to handle.

***GUIDANCE #2**: ensure all developers know who the partition managers are.*

Partition managers develop, over time, a comprehensive understanding of their owned partitions, ensuring that future modifications are done with complete knowledge of the ramifications of modifying any software within the partition.

Different operating systems (and microprocessors) have varying capabilities in terms of enforcing strict separation between components. For example, a small, real-time operating system may not make use of a computer's memory management unit at all; multiple software applications cannot be protected from each other, and the operating system itself is at risk from flaws in application code. These *flat memory model* operating systems are not suitable for complex, partitioned software systems. General purpose desktop operating systems such as Linux and Windows employ *basic memory protection*, in which partitions can be assigned processes that are protected from corruption by the memory management unit, but do not make hard guarantees about availability of memory or CPU time resources.

A family of operating systems, such as the INTEGRITY real-time operating from Green Hills Software, provide strict partitioning of applications in both time and space. A damaged application can not exhaust system memory, operating system resources, or CPU time because the faulty software is strictly limited to an assigned quota of critical resources. The quota affects literally all memory in use, including heap memory for the C/C++ runtime, memory used for process control blocks and other operating system objects, and processes' runtime stack memory. In addition, the partitioning policies provide strict quotas of execution time and strict control over access to system resources such as I/O devices and files. A more rigorous partitioning of applications at the operating system level ensures that the benefits of partition management policies used in the development process are realized during run-time.

***GUIDANCE #3**: if possible, use an operating system that employs true application partitioning.*

3 CHANGE MANAGEMENT

A software project may be robust and reliable at the time of its first release, only to endure *change rot* over ensuing years as new features, not part of the original design, are hacked in, causing the code to become difficult to understand, maintain, and test. Time to market demands exacerbate the problem, influencing developers to make hasty changes to the detriment of reliability. Therefore, an extremely important aspect of maintaining reliable software over the long term is to utilize an effective change management regimen. Some fundamentals of quality change management, such as the employment of configuration management systems to control and record changes and manage code branches and releases, are assumed and not covered in this paper.

3.1 Peer Reviews

Many rigorous development processes involve the use of peer code reviews. A common peer code review sequence consists of the code author developing a presentation describing the code change followed by a face to face meeting with one or more developers and development managers involved in the project. The developer presents the software design in question, and the others try to poke holes in the code. These meetings can be extremely painful and time consuming. Audience members sometimes feel compelled to nitpick every line of code in order to demonstrate their prowess.

***GUIDANCE #4**: use asynchronous code reviews with email correspondence instead of face-to-face meetings.*

Partition management drastically reduces the time required for code reviews since the code experts are almost always the one modifying their own partitions. Debates regarding design decisions are usually avoided. In addition, we advocate avoiding face-to-face peer reviews except as absolutely necessary. The partition manager applies the changes to a local copy of the software, selects a suitable peer to review the changes, and then makes a review request via email. Soon after, but at a time convenient for the reviewer, the reviewer reviews the code differences at his desk, and then sends back comments via email to the author. When the author receives an email indication that the change is approved, the software is committed to the configuration management system. If a reviewer rejects a modification, the author must correct any discovered flaws or, if he disagrees with the assessment, appeal to the common manager to referee. The configuration management system must provide for the ability to specify the reviewer's user identification as part of the commit comment. For example, CVS allows a script to be run during a commit; the script is provided the commit comment which is parsed for the user identification. If a valid user identification is not found, the CM system rejects the commit. Thus, the CM system can be used to automate the enforcement of a code review policy. Without such an automated system, there can be no guarantee that a developer will not commit a change that has not been properly vetted.

***GUIDANCE #5**: use the CM system to automate enforcement of peer reviews for every modification to critical code.*

Recording the reviewer's identification in the CM system also provides an electronic paper trail for auditors.

Another advantage of partitioning is the ability to minimize process requirements across the system. In any large software project, there is a continuum of criticality amongst the various pieces of code. By way of example, let us consider an exciter laser system used in semiconductor manufacturing. The laser itself is controlled by a highly critical, real-time software application. If this application faults, the laser in turn may fail, destroying the semiconductor. In addition, the system contains a communications application that uses CORBA over TCP/IP to receive commands and to send diagnostic data over a network. If the communications application fails, then the system may become unavailable or diagnostic data may be lost, but there is no possibility for the laser to malfunction. If both applications were built into a single, monolithic system in which all code executes in the same memory space, then the entire software content must be developed at the highest levels of quality and reliability. If the applications are partitioned, however, the non-critical communications application development can be subjected to a lower level of rigor, saving time to market and development cost.

***GUIDANCE #6**: apply a level of process rigor, e.g. code reviews and other controls, that is commensurate with the criticality level of the partition.*

Obviously, we do not advocate a free-for-all on code partitions that are not considered critical; management should use judgment regarding which controls to apply to various software teams. By reducing the process controls in non-critical applications, time to market for the overall system can be improved without jeopardizing reliability where it counts.

3.2 Improving Efficiency of the Build Cycle

When software changes rapidly, the efficiency of the build process becomes a critical component of developer efficiency. Complex software projects are often characterized by complex build processes, where the software not only takes a long time to recompile from scratch, but also may require many recompiles in order to exercise different production configurations. For example, a software system may have a "production" build, where compiler optimizations are fully enabled and the software is configured for maximum speed and reliability; a "debug" build where the system has debugging information enabled so that developers can most easily debug the software; and a "checked" build where the system turns on additional sanity checks that may drastically reduce performance but increase the probability of finding unusual problems such as RAM hardware failures.

During development, it may not be practical for a developer to build all configurations to test a change. Therefore, an *autobuild* system should be used. When a change is committed to the CM system, one or more dedicated build computers update their local checkouts of the software and rebuild all configurations. When a build fails, the autobuilder sends an automated email to the partition manager(s) of the affected partition(s). An email is also sent to the person in charge of the build system. The autobuild system ensures that erroneous changes causing build failures are immediately detected, before they affect other developers.

***GUIDANCE #7**: use an autobuild system to quickly detect changes that break system builds.*

With x86-based PCs and servers besting several GHz, developers are enjoying faster builds than ever before. However, as software complexity has grown in concert, build times remain an important factor relating to developer efficiency. Yet there are a couple of methods to reduce the effect of build times on developer productivity.

GUIDANCE #8: always ensure a developer has at least two development projects to work on at all times.

When a change is made and a long build is started, there is no excuse for a developer to be waiting for that build to complete. A developer should always have a secondary project to work on during inevitable breaks, such as those caused by waiting for builds, peer reviews, or the many other reasons why a foreground project is delayed. Ensuring that a developer has multiple projects to work on at all times is, ultimately, the responsibility of the developer's management. However, a developer naturally considers down time waiting for a build to complete as a normal mode of operation where the developer has no choice but to take a break. Therefore, it is important to teach the developer to proactively request more work in the case he finds himself blocked on all fronts.

Another technical solution to the waiting build problem is to reduce build times by bringing to bear the full power of corporate computer horsepower. A typical development site may employ many developers, working on the same or different software projects. There may be dozens, if not hundreds, of PCs on the desks of these developers. Sadly, these PCs typically spend the majority of their time idle.

***GUIDANCE #9**: employ distributed builds to maximize computer utilization and improve developer efficiency.*

Some compiler vendors provide distributed build capability. Do not confuse, however, something like parallel "make" [2] that can parallelize build operations on a single, typically multiprocessor, host computer. To truly scale, the parallel build system must be able to interrogate PC resources throughout a site, locate machines that are under-utilized, and migrate the files required to accomplish distributed builds to these available resources. Ideally, a distributed build system should not require significant configuration. The only required piece of configuration information for the tool is the knowledge of which machines on the network should be candidates to share in the workload. Distributed builds greatly reduce build times, therefore shortening the build-edit-debug cycle that is so crucial to developers' productivity on a daily basis. Another beneficial side effect of distributed builds is the

potential to reduce developer capital expenses by better utilizing compute resources throughout a site.

4 Coding Standards

Most high assurance development processes espouse the use of a coding standard that governs how developers write code [3]. Some of them go further to recommend or require specific rules be included in the coding standard. The goal of the coding standard is to increase reliability by promulgating intelligent coding practices. For example, a coding standard may contain rules that help developers avoid dangerous language constructs, limit complexity of functions, and use a consistent syntactical and commenting style. These rules can drastically reduce the occurrence of flaws, make software easier to test, and improve long term maintainability.

***GUIDANCE #10**: develop and deploy a coding standard that governs software development of all critical partitions.*

It is not uncommon for a coding standard to evolve and improve over time. For example, the development team may discover a new tool that can improve code reliability and recommend that management add a requirement that this tool be used during the development process.

It is also not uncommon to see a coding standard consisting of guidance rules whose enforcement is accomplished primarily with human code reviews. Developing a new coding standard with dozens of rules that must be verified manually is a sure way to reduce developer efficiency, even if it is increasing the reliability of the code.

***GUIDANCE #11**: maximize the use of automated verification of the coding standard; minimize the use of manually verified coding rules.*

Although some coding standard rules are necessarily language-specific (and omitted in this paper in order to avoid inapplicability to a large percentage of readership), there are some universally or almost universally applicable rules that should be a part of a high quality coding standard.

***GUIDANCE #12**: prohibit compiler warnings.*

Compilers and other tool chain components (e.g. the linker/loader) often emit warnings, as opposed to halting a build with a fatal error. Warnings are an indicator to the developer that a construct may be technically legal but believed to be dangerous (sometimes the cause of subtle bugs). To ensure that warnings are not intentionally or accidentally ignored by developers, tell the compiler to treat all warnings as errors. Many compilers have such an option.

***GUIDANCE #13**: take advantage of the compiler's strictest language settings for safety and reliability.*

Compilers also tend to provide a variety of strictness levels in terms of language standard interpretation. In addition, some compilers are capable of warning the developer about constructs that are technically legal but dangerous. For example, the Motor Industry Software Reliability Association (MISRA) has published guidelines for the use of the C language in critical systems [4], and some compilers can optionally enforce some or all of these guidelines that essentially subset the language by excluding constructs believed to lead to unreliable software. Some MISRA guidelines are advisory and may yield warnings instead of errors; once again, if the MISRA rule is enabled, the compiler should be forced to generate a fatal build error on any noncompliant construct.

It goes without saying that organizations should strongly weigh the diagnostic capability of a compiler when selecting such an important tool.

***GUIDANCE #14**: if a coding standard rule cannot be fully enforced at compile time, try to enforce it in a post-compile phase.*

A good example of this would be a code checking tool that detects when a NULL pointer is passed to a function, defined in another source file, that unconditionally dereferences the pointer. A traditional compiler is unable to detect this coding error since the compiler's input is a single source file. An integrated code checking tool would take the output from compiling the file containing the call passing the NULL pointer and the output from the file containing the function definition, and detect this inter-module coding error. Although inter-module code checking tools can take a long time to run through a large project's code base, the number of coding errors that can be found above and beyond a normal compiler's purview may be well worth the expense. At least these code checking tools should be executed occasionally if not on every build.

4.1 Improper Symbolic Resolution

Another family of inter-module coding errors involves the accidental use of a function definition that is not suitable for the reference. For example, if a compiler does not mangle the symbolic name of a function using the function's signature (common in languages such as C), then a function call may be resolved with a definition whose parameter or return types do not match. Or a variable reference of one type may be resolved with a definition whose type is incompatible. For example, consider the following C function definition and code reference, each located in a separate source file:

File1:

```
void read_temp_sensor(float *ret) {
   *ret = *(float *)0xfeff0;
}
```

File2:

```
float poll_temperature(void) {
   extern float read_temp_sensor(void);
         return read_temp_sensor();
      }
```

The above code fragments are perfectly legal ANSI/ISO C. However, this software will fail since the reference and definition of read_temp_sensor are incompatible (the former is written to retrieve the return value of the function

while the latter is written to return the value via a reference parameter).

***GUIDANCE #15**: enforce valid resolution of code references to definitions.*

One obviously poor coding practice illuminated above is the use of an extern function declaration near the code containing the reference. Although ANSI C requires a prototype declaration, the scope of this declaration is not covered by the specification. MISRA attempts to prevent this coding pitfall by not allowing function declarations at function code level. However, the following code fragment would pass this MISRA test yet fail in the same manner as the preceding example:

```
extern float read_temp_sensor(void);
float poll_temperature(void) {
        return read_temp_sensor();
}
```

Another method would be to combine the MISRA rule with a second rule that disallows the use of prototype declarations anywhere other than in header files. This certainly makes the error less likely, yet still falls short of guaranteeing compliance to the coding standard (the header file containing the declaration may not be used in the source file containing the incompatible definition).

There is really only one way to guarantee that the declaration and definition match: detect incompatibilities post-compile, such as at link-time. When compiling the aforementioned code fragment, the compiler can insert into its output file some marker (such as a special symbol in the symbol table or a special relocation) that describes the signature of the return type and parameter types used in a function call. When the function definition is compiled, the compiler outputs the signature for the definition as well. At link time, when the final executable image is being generated, the linker/loader compares the signature for same-named functions and generates an error if any incompatible signature is detected. This additional checking should add negligible overhead to the build time (the linker already must examine the references of functions in order to perform relocation) yet guarantees enforcement of the coding standard rule and therefore improves reliability and quality of the resulting software.

Another example of improper symbolic resolution relates to the unintended use of exported library definitions. Libraries are often used to collect code modules that provide a related set of functions. For example, most operating systems come with a C library, e.g. libc.so, that provides support for the C runtime, including string manipulation, memory management, and console input/output functions. A complex software project is likely to include a variety of project-specific libraries. These libraries export functions that can be called by application code. A reliability problem arises due to the fact that library developers and application developers may not accurately predict or define a priori which interfaces are exported by the library. The library may define globally visible functions intended for use only by other modules within the library. Yet once these functions are added to the global namespace at link time, the linker may resolve references made by applications that were not intended to match the definitions in the library. For example, consider an application which makes use of a "print" function. The application developer envisioned the use of a print library provided by the printer management team. However, the font management team created a library, also used by the application developer, that provides a set of font manipulation functions. The font management team defined a print function intended for use by other modules within the font management library. However, because there often is no facility for limiting the namespace of libraries, the font library's print function was inadvertently used by the linker to resolve "print" references made by the application developer, causing the system to fail.

Therefore, GUIDANCE #15 may need to be enforced with something other than the compiler's front end. For example, a tool can be used to hide library definitions so that they are used by the linker when resolving intra-library references but ignored when resolving extra-library references. The Windows platform employs user-defined library export files to accomplish this separation [5]. When creating Windows DLLs, developers specify which functions are exported. Functions not included in the export file will not be used to resolve application references.

Note that some high level languages (such as Ada) do a better job of automatically enforcing type consistency and name spacing than other languages (such as C). Language choice could very well make guidance #15 trivial to enforce.

4.2 Complexity Control

Much has been published regarding the benefits of reducing complexity at the function level. Breaking up a software module into smaller functions makes each function easier to understand, maintain, and test [6]. One can think of this as *meta-partitioning*: applying the aforementioned software partitioning paradigm at a lower, programmatic, level. A complexity limitation coding rule is easily enforced at compile time by calculating a complexity metric and generating a compile-time error when the complexity metric is exceeded. Once again, since the compiler is already traversing the code tree, it does not require significant additional build time to apply a simple complexity computation, such as the popular McCabe complexity metric. Because the compiler generates an actual error pointing out the offending function, the developer is unable to accidentally create code that violates the rule.

***GUIDANCE #16**: use automated tools to enforce a complexity metric maximum, and ensure that this maximum is meaningful (such as a McCabe value of 20).*

Adopting a coding standard rule that allows a McCabe complexity value of 200 is useless; most any legacy code base will be compliant despite having spaghetti-like code that is hard to understand, test, and maintain. The selection

of a specific maximum complexity value is open to debate. If an existing code base is well modularized, a value may be selected that allows most of the properly partitioned code to compile; future code will be held to the same stringent standard. When applying the complexity metric to a large code base that has previously not been subjected to such an analysis, it is likely that a small number of large functions will fail the complexity test. Management then needs to weigh the risk of decomposing a large function with the risk of changing the code at all. Modifying a piece of code that, while complex, is well exercised (proven in use) and serves a critical function may very well reduce reliability by increasing the probability of introducing a flaw. The complexity enforcement tool should provide a capability to allow exceptions to the complexity enforcement rule for specific functions that meet this profile. Exceptions, of course, should always be approved by management and documented as such. The coding standard should not allow exceptions for code that is developed subsequent to the adoption of the coding rule. These types of coding standard policies conform to their spirit while maximizing efficiency, enabling them to be employed effectively in legacy projects.

5 Software Testing and Verification

Effective testing is well known to be one of the best mechanisms to assure that software is reliable. Therefore, it is an important component of many high assurance development standards and guidance documents, such as that promulgated by the U.S. Food and Drug Administration [7]. Testing includes functional testing, regression testing, performance testing, and coverage testing. In the realm of functional testing we have fault-based testing, error-based testing, white-box testing, and black-box testing. It is left as an exercise to the reader to explore the various types of testing and their relative advantages. Our guidance is more concerned with the integration of testing methodology into the development process in order to maximize its value.

Organizations that do not follow a rigorous development process often resort to ad-hoc testing that is often an afterthought when most of the software has already been written. Organizations that follow a rigorous process often focus testing during a release process, again after much of the software has been written.

***GUIDANCE #17**: the testing system should be running 24x7.*

If a testing system is only run on demand, occasionally, or only during a release process, then errors which can be detected by the testing system tend to go unnoticed for an unnecessarily long period of time. When a flaw is discovered, the developer has a much harder time trying to remediate it than if the flaw was introduced the previous day. In some cases, the developer may have even moved on to another project if not another company, leaving someone else to try and learn the code and fix the flaw. Fixing flaws discovered by the testing system should be prioritized higher than anything other than emergency customer support issues; keeping the system running cleanly at all times guarantees that test system failures are almost always new failures that have not been examined by anyone else and need immediate attention.

***GUIDANCE #18**: the testing system should run on the development version as well as active shipping versions.*

This is a corollary to GUIDANCE #17 but important enough to be stressed: when a testing system is used throughout the development process, developers are forced to keep the product in a working state at all times. Software projects that only move to rigorous test after a code freeze are subjected to test phases that last longer overall because developers must wrestle with problems inserted throughout months of development time. When a product is always working, a code freeze leads directly to final quality assurance testing, saving time to market. If a developer can not develop code in a manner that prevents the product from failing, then a private branch can be used as long as it is not allowed to live too long; integrating old code branches that have drifted far from the trunk often causes unforeseen conflicts that affect the efficiency of the entire development team.

***GUIDANCE #19**: the testing system should be able to effectively test a software project in less than one night.*

A testing system that takes too long to run tends to become underutilized if not completely ignored. Developers should be able to quickly validate a change overnight before committing it to the master version of the project. In addition, the automated tests running 24x7 on dedicated testing compute farms can detect flaws very quickly so they can be corrected while the understanding of the recently added code is still fresh in the developer's mind. It is reasonable to have more tests that can run in one night; however longer runs should compete at a lower priority for computing resources or be run only on demand or at longer intervals during the development process. The nightly test run will almost always be good enough to detect flaws entered during development.

***GUIDANCE #20:** it should be trivial to determine when a test run has succeeded or failed; a failed test should be trivial to reproduce.*

Tests should be written such that output is generated only when an error is detected. A clean test is one without any output. At worst, the output should be less than a page long. Too often, testing systems generate voluminous output, making it difficult for developers to quickly ascertain the status of the test run. Test output that is difficult to quickly evaluate tends to be ignored and ineffectual.

When a test fails, the exact state of the software system and any inputs or process that must be used to reproduce the discovered error should be clearly displayed within the test output. If the developer is unable to efficiently reproduce a test failure, the test system will tend to be ignored. Reproducibility is the key to maximizing the rate at which developers can remediate flaws discovered by the testing

system and bring a reliable software product to market faster.

6 Conclusion

High assurance development processes, such as that encouraged by ISO/IEC quality standards as well as regulatory bodies that govern the use of software in safety and security systems, is generally believed to result in higher reliability software. However, an organization that follows these processes to the letter is likely to get to market later than an organization that develops the same product without the overhead of following these quality standards. In this paper we have presented a set of guidance recommendations, part of a process that has been proven in use for 24 years at one of the leading reliability-critical software development organizations in the world, and have argued how these controls can actually increase developer productivity and reduce time to market through the use of methodologies that find software flaws faster and reduce developer dependencies on each other and on productivity-killing bureaucracy.

References

[1] Robert Parker. *Are Vendors Doing Enough To Improve Software*? Optimize Magazine; http://www.optimizemag.com/issue/009/squareoff.htm

[2] Morgan Herrington. *Optimizing Build Times Using Parallel "make"*. http://developers.sun.com/solaris/articles/parallel_make.html

[3] RTCA (1992). *DO-178B, Software Considerations in Airborne Systems and Equipment Certification*.

[4] The Motor Industry Software Reliability Association. *MISRA-C:2004 Guidelines for the use of the C language in critical systems*. October 2004.

[5] Microsoft Development Network. *Working with Import Libraries and Export File.* http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vccore/html/_core_working_with_import_libraries_and_export_files.asp

[6] Arthur H. Watson and Thomas J. McCabe. *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*. http://www.mccabe.com/pdf/nist235r.pdf

[7] U.S. Food and Drug Administration, Center for Devices and Radiological Health. *General Principles of Software Validation*. http://www.fda.gov/cdrh/comp/guidance/938.html

Contact

David Kleidermacher is chief technology officer at Green Hills Software where he has been developing real-time operating systems and tools for embedded systems for the past fifteen years. David has a bachelor of science in computer science from Cornell University. Contact David at davek@ghs.com.

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 : URL : www.ghs.com

I-Logix
Contact: Martin Stacey
1 Cornbrash Park, Bumpers Way, Chippenham, Wiltshire, SN14 6RA, UK
Tel: +44-1249-467-600 Fax: +44-1249-467-610
Email : info_euro@ilogix.com URL : www.ilogix.com

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