ADA USER JOURNAL

Volume 28
Number 3
September 2007

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

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

Reprinted Articles

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

Editorial

As the new editor-in-chief of the Ada User Journal, I am pleased to start by thanking Tullio Vardanega for his effort and accomplishments in his five years of service. During this period, Tullio was able to uncover diverse sources of interesting content, providing regular and worthwhile material for us. Well done, Tullio! I am also pleased to salute all the Journal authors and readers. I have now the (difficult) task of taking over, and I am confident that you will help me in making the Journal continue in the right path.

If you do not mind, I would like to talk about another, more important, change that took place in Ada-Europe. During our flagship conference, last June in Geneva, the General Assembly witnessed the end of the six years of service of Erhard Plödereder as the President of Ada-Europe. At the same Assembly, Tullio was unanimously voted for the position. I would like to wish Tullio a great success in his new role, and congratulate Erhard for the Ada-Europe accomplishments during his tenure. I am glad that Erhard will continue in the Ada-Europe board, as a member at-large, providing us with his vast experience and knowledge.

Coming back to the Journal, I have to apologise for the delay in the production of the issue you have in your hands. I have just started to understand the effort that it is necessary to build each individual issue. It would be an impossible job, if not for the commitment of the editorial team. My deepest and sincere thanks to Santiago Urueña, Dirk Craeynest and Jorge Real for their support. As for its contents, the news and calendar section form an important part of the issue. In these days of vast, overwhelming, information, the job of data mining is a deeply important, but difficult one. Therefore I strongly recommend you to read the sections that Santiago and Dirk have prepared. Also please note the forthcoming events section, where you will encounter the call for participation for SIGAda 2007 and the call for papers for Ada Europe 2008. The technical part of the issue provides three papers from the Industrial Track of the Ada Europe 2007 conference. In the first paper, Javier Miranda of the University of Las Palmas de Gran Canaria, Spain, describes some of the enhancements currently being applied to the GNAT technology to support certification of object-oriented Ada code for high-integrity systems. The second paper, by Paul Black of the National Institute of Standards and Technology, USA, presents the SAMATE project, which addresses evaluation measures and methods and tools for software assurance. The last paper of this issue is by a group of authors from Rapita Systems and Hawk Mission Systems, both in UK, reporting the results of a study for the reduction of worst-case execution time in the operational flight program of the Hawk mission computer. I am sure you will enjoy reading them as much as I did.

To close this editorial, let me shed some light into the near future. As mentioned in the previous issue, an effort was being made to obtain permission to publish the proceedings of the 13th International Real-Time Ada Workshop. I am very pleased to inform you that this has been granted and we will publish the staggered proceedings of this important workshop in the next issues of the Journal. I am also happy to inform that we are taking similar steps to obtain permission to publish selected contributions from the AdaCore Gem of the Week series. Therefore, the future looks promising; and I challenge all of you to play a part building it.

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

News

Santiago Urueña

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

Contents

Ada-related Organizations

Availability of Ravenscar Guide

From: Jim Moore <moorej@mitre.org>
Date: 9 July 2007 16:33:50 GMT+02:00
To: WG9 Participants
Subject: Availability of Ravenscar Guide

ISO Central Secretariat has finally implemented the long-requested free availability of the Ravenscar Guide, ISO/IEC TR 24718:2005. It, and other freely available standards, can be found at:

http://isotc.iso.org/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm

James W. Moore, CSDP, F-IEEE

The MITRE Corporation

AIs entering Editorial Review

From: Pascal Leroy <pascal.leroy@fr.ibm.com>
Date: 28 August 2007 09:18:22 GMT+02:00
To: tullio.vardanega@math.unipd.it
Subject: AIs entering Editorial Review

To WG9 HODs and Officers

In compliance with resolution 44-4 of WG9, this message is to inform you that the AIs listed below have entered Editorial Review, and are intended to be submitted to WG9 for approval at the next meeting (meeting #53 in Washington, DC, USA). It is expected that HODs and liaison representatives would take this opportunity to circulate these AIs for comments within their respective organizations, and return comments to the ARG as soon as feasible.

The editorial review period ends on September 21, 2007. Note that substantive comments received after this date would probably cause the corresponding AIs to be removed from the list submitted to WG9, as there would not be enough time left to properly answer the comments.

The AIs can be found on-line at http://www.ada-auth.org/AI05-SUMMARY.HTML.

AI05-0002-1/03 2007-06-11 — Unconstrained arrays and C interfacing
AI05-0008-1/04 2006-12-13 — General access values that might designate constrained objects
AI05-0017-1/03 2007-06-17 — Freezing and incomplete types
AI05-0019-1/03 2007-06-15 — Primitive subprograms are frozen with a tagged type
AI05-0024-1/04 2007-06-18 — Run-time accessibility checks
AI05-0028-1/05 2007-06-18 — Problems with preelaboration
AI05-0035-1/03 2007-06-18 — Inconsistences with pure units
AI05-0037-1/01 2007-01-22 — Out of range <> associations in array aggregates
AI05-0040-1/02 2007-06-18 — Limited with clauses on descendants
AI05-0043-1/01 2007-06-15 — The Exception_Message for failed language-defined checks.
AI05-0046-1/02 2007-06-18 — Null exclusions must match for profiles to be fully conformant
AI05-0055-1/02 2007-06-12 — Glitch in EDF protocol
AI05-0056-1/02 2007-06-17 — Wrong result for Index functions

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 12 — Ada-Belgium Technical Presentation

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Subject: "Ada at Barco Avionics" presentation now on Ada-Belgium site
Date: Sat, 16 Jun 2007 07:16:12 +0000 UTC
Organization: Ada-Belgium, c/o Dept. of Computer Science, K.U.Leuven
Newsgroups: comp.lang.ada,fr.comp.lang.ada,be.comp.programming,nl.comp.programmeren

Post-event announcements
Ada-Belgium recently organized
a technical presentation by
Ludovic Brenta of Barco Avionics,
Belgium
Ada at Barco Avionics:
history, coding standards, and products
Tuesday, June 12, 2007

Mini-report by a participant

On the 12th of June, Ludovic Brenta gave a very interesting talk to Ada-Belgium on the use of Ada at Barco. Barco is a leading European supplier of displays and related technologies. Ada is used in Barco's avionics products, predominantly cockpit display and control-display units. As in all computing the processing power of these devices has risen and is now enough to support reasonable amount of software. Typically these devices contain some sort of display software, communications, exception logging and self test, though the latest devices have enough processing power to allow other applications to run in a time sharing configuration.

All of this must be certifiable to the prevailing standard — DO 178B — and almost all is written in Ada — a few lines of machine code and some C is used in some cases. The requirements of certification were outlined and the resulting constraints on the development team were described. Barco has developed a coding standard to enable certifiable software to be written and the rationale of the standard was explained.

Presentation available online

The slides of this technical presentation are available on-line here: <http://www.cs.kuleuven.be/~dirk/ada-belgium/events/07/070612-abga-event-aba.pdf> "Ada at Barco Avionics: history, coding standards, and products", June 2007 (Adobe Portable Document Format, PDF, 2296 KB).

[See also "Jun 12 — Ada-Belgium" in AUJ 28-2 (Jun 2007), pp.70–71. —su]

Jun 25–29 — Ada-Europe 2007

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.ac.be>

To: Ada-Europe-attendees@cs.kuleuven.be
Date: Sun, June 17, 2007 4:19 pm
Subject: Press Release — Reliable Software Technologies, Ada-Europe 2007

FINAL Call for Participation
12th International Conference on Reliable Software Technologies — Ada-Europe 2007
25 – 29 June 2007, Geneva, Switzerland
http://www.ada-europe.org/conference2007.html

Press release:

Ada-Europe Conference on Reliable Software Technologies

International experts meet in Geneva

Geneva (17 June 2007 16:00) — Ecole d'Ingénieurs de Genève together with and sponsored by Ada-Europe, and in cooperation with ACM's Special Interest Group in Ada, organize this year the "12th International Conference on Reliable Software Technologies — Ada-Europe 2007" from 25 to 29 June in Geneva, Switzerland.

The conference offers 8 tutorials, a full technical program of refereed papers, a collection of industrial presentations reflecting current practice and challenges, four invited speakers, an industrial exhibition, and a social program.

The 8 excellent tutorials on Monday and Friday cover a broad range of topics: An Overview of Model Driven Engineering, Correctness by Construction — a UML2 Profile Enforcing the Ravenscar Computational Model, Verification and Validation for Reliable Software Systems, Object-Oriented Programming in Ada 2005, Security by Construction, Synchronous Design of Embedded Systems — the Esterel/Scade approach, Building Interoperable Distributed Applications with PolyORB, and Situational Method Engineering — Towards a Specific Method for each System Development Project.

The technical program presents 18 fully refereed and carefully selected papers on the latest research, including new tools, applications and industrial practice and experience, and a collection of 7 industrial presentations reflecting current practice and challenges. Springer Verlag publishes the proceedings of the conference, as LNCS Vol. 4498.

Four international experts present invited lectures on the topics: Challenges for Reliable Software Design in Automotive Electronic Control Units, Synchronous Techniques for Embedded Systems, Perspectives on Next Generation Software Engineering, and Observation Rooms for Program Execution Monitoring.

The exhibition opens in the mid-morning break on Tuesday and runs continuously until the end of the afternoon break on Thursday. The exhibitors include the following vendors: AdaCore, Aonix, Ellidiss Software (TNI-Europe), Green Hills Software, the Hibachi project, Praxis, Programming Research BV, Rapita Systems Ltd, and Telelogic.

The social program includes on Tuesday evening a visit of, and reception at, the building of the World Meteorological Organization (WMO, agency of the United Nations), and on Wednesday evening an aperitif in the History of Sciences Museum followed by the conference banquet at the restaurant La Perle du Lac, close to the Leman Lake.

The conference takes place at the Engineering School of Geneva (Ecole d'Ingénieurs de Genève-EIG), 4 Rue Prairie, in the center of Geneva. The full "Advance Program" is available on the conference web site and directly at <http://adae2007.eig.ch/docs/avp.pdf>. Registration is still open.

Latest updates:

- The "Final Program" is available on the conference web site <http://www.ada-europe.org/conference2007.html> and directly at <http://adae2007.eig.ch/docs/finalp.pdf>.

- Check out the 8 tutorials in the advance program and at <http://adae2007.eig.ch/tutorial.html>.

- The proceedings, published by Springer Verlag as Lecture Notes in Computer Science Vol. 4498, are ready and will be distributed at the conference. More info is available at <http://www.springer.com/978-3-540-73229-7>.

- Registration fees are very reasonable and the registration process is easy: fill out the 1-page form at <http://adae2007.eig.ch/docs/register.pdf> and fax it to the conference secretariat. Don't delay!

- For the latest information consult the conference web site.

SPARK-related events

September 2007

LASER Summer School on Software Engineering

September 9th-15th 2007, Elba, Italy

Rod Chapman will be presenting a talk about SPARK and its verification tools.

AdaUK Conference, September 25th, Manchester, UK

SPARK team will be exhibiting, and presenting both a technical paper and a vendor track presentation at this event.

October 2007

Embedded Systems Show, October 18th, Birmingham NEC, UK

Rod Chapman will be presenting a 1-hour tutorial on programming language design issues and static verification for dependable systems, as part of the IET Technical Conference "Design of Dependable Systems" Track.

IET Conference on System Safety, 22nd-24th October, London, UK

SPARK Team are presenting a tutorial on Correctness by Construction on Monday 22nd October.

November 2007

ACM SIGAda 2007, 4th-9th November, Washington DC, USA

Rod Chapman will be presenting both a tutorial on "Security by Construction" and one of the key-note speeches at SIGAda this year.

IMechE Software Reliability Seminar, 20th November, London, UK

Hibachi Events

From: Tom Grosman <grosman@aonix.fr>
Newsgroups: eclipse.tools.adt
Subject: Hibachi Presentation at AdaEurope 2007
Date: Tue, 19 Jun 2007 18:18:26 +0200

There will be a presentation of Hibachi at the Ada-Europe conference in Geneva next week (June 26) during the vendor sessions. In addition, I will be manning a Hibachi booth in the Exposition hall.

The purpose is to introduce the Ada community to the Hibachi project and more generally Eclipse and the Eclipse community, to spread awareness of the project in order to build up the community (especially amongst institutions), and to answer questions and receive feedback

For information on the conference, see http://adae2007.eig.ch/.

From: Tom Grosman <grosman@aonix.fr>
Subject: Hibachi Workshop at SigAda in November
Date: Tue, 19 Jun 2007 19:06:27 +0200
Newsgroups: eclipse.tools.adt

The SigAda program committee has accepted a proposal for a workshop on Hibachi at SigAda in Washington DC this year. The workshop is scheduled for the evening of November 8. See http://www.acm.org/sigada/conf/sigada2007/workshops.html for workshop details and http://www.acm.org/sigada/conf/sigada2007/ for conference details.

Tom Grosman

Hibachi Project Lead

Ada and Education

Ada 2005 & Java Syntax

From: Maciej Sobczak <maciej@msobczak.com>
Newsgroups: comp.lang.ada

Subject: Re: Java-Ada 2005 Syntax / Language Features Comparisons
Date: Wed, 08 Aug 2007 23:44:11 −0700

> Is anyone aware of a reference card or short document that shows equivalent Ada syntax and language features with those of Java.

> Students could use this to understand data structure concepts written in a book using Java, and then implement these concepts in code using Ada 2005. These students are CS majors and will have already taken a course in Ada.

If they are CS majors, then they should be able to think in terms of abstracts and should not need such primitive cheat-sheets for 1:1 translations between languages.

I would even go further and say that the whole idea is broken at the start. Java is reference-oriented whereas Ada is value-oriented, which has significant consequences in how high-level concepts like composition and aggregation are expressed in code — this goes much further than syntax differences. Teaching people to recode some stuff using "syntax equivalents" is a Bad Idea.

From: Anilkumar.T <Anilkumar.Thimmaiah@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Java-Ada 2005 Syntax / Language Features Comparisons
Date: Thu, 09 Aug 2007 09:46:51 −0000

You can read this [comparison with] The Steelman

http://www.adahome.com/History/Steelman/steeltab.htm

Date: Thu, 09 Aug 2007 13:15:49 +0200
From: Georg Bauhaus <bauhaus@futureapps.de>
Newsgroups: comp.lang.ada
Subject: Re: Java-Ada 2005 Syntax / Language Features Comparisons

What kind of book is this? Would it be impractical to just use an Ada book?

Key notions will include, with their syntax,

Java class <-> Ada package + tagged type (side note: JVM classes use tag fields, too!)

Java packages <-> Ada package hierarchies

Java subtypes <-> Ada packages and derived types

Java public/protected/private <-> Ada public/private + visibility rules + nesting

Java low level concurrency building blocks <-> Ada built in concurrency features

We have tried to collect a few hints in http://en.wikibooks.org/wiki/Ada_Programming/Object_Orientation

Approaching the subjects from a conceptual point of view seems like a good opportunity to me. You can then demonstrate, for example, where and when values are better than references, see the benefits of a well defined base type system, etc. In particular when the students have already taken a course in Ada.

Syntax only transformations are indeed prone to financial and technical disaster. They can be dangerous. There is enough anecdotal evidence already. But I'm not sure this fits the OP's motivation?

One more anecdote: A programmer used Java for programming but wrote identifiers such as

performThisActionOnThingWithThatConstraint(equallyLengthyArgumentValue, …);

It looked like the programmer had done some vanilla Scheme programming before and was mathematically skilled. Would syntax charts for plain Scheme <-> Java have helped at all? I doubt it. However, studying the first chapters of any O-O methods book such as the ones by Booch would have helped as these will inevitably make you notice the method of finding objects by looking at function names…

From: Mike McNett <michael.mcnett@usma.edu>
Newsgroups: comp.lang.ada
Subject: Re: Java-Ada 2005 Syntax / Language Features Comparisons
Date: Thu, 09 Aug 2007 05:33:52 −0700

- I already have the Steelman reference and will use a couple snippets from it.

- While I've considered using a Data Structures book that uses Ada 2005, I've decided against it for several reasons.

- I agree that having "primitive cheat-sheets" is not best for all NEW CS majors, there are a handful of students whose learning styles call for something like this. If a student doesn't want (or need) to refer to it, they certainly don't have to.

- The next course they take uses Java, although we don't teach them Java — it is expected that they do some self-study to learn the language. This means the following: CS1 teaches problem solving using Ada; CS2 teaches data structures using Ada (but using a Java data structures book); CS3 is Advanced Programming Concepts (using Java) that focuses quite a bit on Design Patterns.

- The book that I'm using in CS2 also introduces students to UML, sequence diagrams, and several other important concepts that we use in the CS3 course. Therefore, this CS2 course is the "hook" I'm using to link their CS1 course to their CS3 course without directly "teaching" a new programming language. The book used in the CS2 course helps me teach the concepts, with the extra benefit of their gaining some basic familiarity with language they will use in their CS3 course.

- I agree with having recode some stuff using "syntax equivalents" is a bad idea. That's why I stated that they understand the concepts from the book and implement them in Ada. I certainly am not advocating that they understand the SYNTAX from the book and code that in "equivalent" Ada SYNTAX.

- The wiki reference will be helpful.

From: John McCormick <mccormick@cs.uni.edu>
Newsgroups: comp.lang.ada
Subject: Re: Java-Ada 2005 Syntax / Language Features Comparisons
Date: Fri, 10 Aug 2007 15:46:04 −0700

Switching languages in the second course is a sure way to create frustrated students. They are barely able to program in any language after only 15 weeks. They still think concretely in that first programming language. Very few students see the abstractions necessary necessary to springboard to a new language — it is almost like they are starting from scratch. I can't imagine that keeping the same language for the second course but teaching it with a book that uses a different language is much better.

We teach Ada in both CS1 and CS2. We teach algorithmic problem solving in CS1 and move to an OO approach in CS2. We switch to Java in CS3 where patterns are the goal. We don't need any "hooks" in CS2 to motivate a link between CS1 and CS3. And we don't just dump Java on them in CS3 - we provide guidance on how "experienced" programmers learn a new language. I think that this guidance provides our students with an important skill for learning in the future. Our students did not fare nearly as well when we just dumped a new language on them in the third course.

From: Mike McNett <michael.mcnett@usma.edu>
Newsgroups: comp.lang.ada
Subject: Re: Java-Ada 2005 Syntax / Language Features Comparisons
Date: Fri, 10 Aug 2007 19:08:00 −0700

During the previous several semesters this CS2 course did the same thing that I am doing now, except the book used in those semesters used C++. From personal experience, it actually worked quite well for both CS and non-CS students. Yes, it is challenging to them when they first start reading the book and try to implement its concepts in a different language. By the middle of the semester, however, they realize that the fundamentals of the languages are quite similar.

The Spring semester will be a good indicator to see if moving to the Java-based book for CS2 helps them in their CS3 course. I should clarify what I meant by us not teaching them Java. We don't

just "dump a new language on them" without any assistance (although that's what my earlier post made it sound). There is plenty of in-class work between the instructor and the students that helps them understand the fundamentals of Java. I'd be very interested in hearing more about the guidance you provide on "… how «experienced» programmers learn a new language."

Our approach described in the earlier post is meant to give them the opportunity to explore Ada more fully in this CS2 course while simultaneously giving them familiarity with a language they will be using in CS3. There is no expectation that they would actually be able to design and implement an application in Java after this CS2 course.

> We teach Ada in both CS1 and CS2. We teach algorithmic problem solving in CS1 and move to an OO approach in CS2. We switch to Java in CS3 where patterns are the goal.

It certainly sounds like our CS1, 2, and 3 courses are quite similar based on this.

Public Ada Courses

From: Ed Colbert <colbert@abssw.com>
Newsgroups: comp.lang.ada
Subject: [Announcing] Public Ada Courses 27–31 Aug in Carlsbad CA
Date: Mon, 23 Jul 2007 20:02:40 −0700

Absolute Software will be holding a public Ada course during the week of 27 August 2007 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.)

Edward Colbert
President
Absolute Software Co., Inc.
Phone: (760) 929-0612
E-Mail: colbert@abssw.com
Website: www.abssw.com

[See also "Public Ada 95 Courses" in AUJ 27-3 (Sep 2006), p.134. —su]

SPARK Training

Public Course Dates for 2007 — UK

Course 1 — "Software Engineering with SPARK"

10th – 13th September 2007 at the Praxis Offices in Bath. Download the booking form.

3rd – 6th March 2008 at the Praxis Offices in Bath.

Course 2 — "Black-Belt SPARK"

18th – 20th September 2007 at the Praxis Offices in Bath. Download the booking form.

11th – 13th March 2008 at the Praxis Offices in Bath.

ARTiSAN Webinar — Developing and Maintaining Ada with UML

From: ARTiSAN Software Tools <info@artisansw.com>
To: Santiago Urueña <santiago.uruena@upm.es>
Subject: Still time to register — Live Webinar —-- Developing and Maintaining Ada with UML
Date: Sat, 08 Sep 2007 19:49:08 +0200

Developing and Maintaining Ada with UML

Presented by ARTiSAN this Webinar will introduce the concept of the UML Ada Profile before looking into the topics of Reverse Engineering Ada code and Forward Generating Ada code from an UML model.

Areas of demonstration will include:

- Reverse Engineering and navigating through the resulting model.

- Forward Generation of Ada code from the UML model.

Date and Time

The webinar lasts for about an hour and will run at the following times:

- Thursday 23rd August — 10am EDT / 3pm BST / 4pm CEST

- Thursday 23rd August — 1pm EDT / 6pm BST / 7pm CEST

For further information and to register for this webinar, visit:
http://www.artisansw.com/news/webinar_details.aspx?webinarID=23

Joining instructions will be emailed through to you separately, approx. 24 hours prior to the webinar.

If the subject is of interest to you, but the times inconvenient, please email mailto:webinarQA@artisansw.com

Ada-related Resources

Archangel Interactive site

From: Luke A. Guest <laguest@abyss2.demon.co.uk>
Newsgroups: comp.lang.ada
Subject: Archangel Interactive site
Date: Sun, 05 Aug 2007 08:19:32 −0700

I've finally put up my new site. I intend on including some Ada tutorials, and I have written the first one (Abstract Types).

http://www.archangeli.co.uk

From: richtmyer@cox.net
Newsgroups: comp.lang.ada
Subject: Re: Archangel Interactive site
Date: Mon, 13 Aug 2007 06:53:30 −0700

I looked at a couple and found them easy to follow and informative.

Hope you keep it going. […]

AdaCore Live Docs

From: AdaCore Live Docs
Date: Friday February 3, 2006
Subject: Live Docs
RSS: www.adacore.com/category/developers-center/reference-library/documentation/feed/

Live Docs provides an up to the minute snapshot of GNAT Pro technology. As new features and improvements are made to GNAT Pro these changes are immediately added to our product documentation and presented here in Live Docs.

GNAT Compilation System:

- GNAT User's Guide for native platforms
- GNAT Reference Manual

GPS:

- Using the GNAT Programming Studio

GtkAda

- GtkAda Reference Manual
- GtkAda User's Guide

PolyORB

- PolyORB User's Guide

Ada-related Tools

Simple components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple components for Ada v2.6
Newsgroups: comp.lang.ada
Date: Sat, 18 Aug 2007 19:31:03 +0200

The library provides implementation of:

Doubly-linked webs and lists
Reference-counted objects and handles to
Parsers
Persistent objects and handles to
Persistent storage and handles to
Storage pools
Sets and maps
Stacks
Strings editing
Tables (containers of strings)
Unbounded arrays

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

Changes to the version 2.5

- Function Is_Empty for doubly-linked lists,
- Functions Erase and Take for both doubly-linked webs and lists
- An example of doubly-linked list use

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

GNU Ada Compiler

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: ANN:[gnuada] New Solaris 10 release.
Newsgroups: comp.lang.ada
Date: Fri, 24 Aug 2007 21:53:09 +0200

We got a new Solaris 10 release. It's based on gcc-4.2.1. Tools and Compiler have been separated:

[http://gnuada.sourceforge.net/ —su]

Since I don't have root access it's just tar's.

[See also same topic in AUJ 28-2 (Jun 2007), pp.74–75. —su]

Interval arithmetic

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Interval arithmetic for Ada v1.5
Newsgroups: comp.lang.ada
Date: Thu, 16 Aug 2007 22:31:20 +0200

Packages for handling intervals in Ada. Interval arithmetical and relational operations are provided for dimensionless and dimensioned intervals:

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

This version is based on measurement units for Ada v2.4. For GNAT Ada compiler users, GPS project files were included in two variants: with and without GTK+ support.

QtAda binding

From: Vadim Godunko <vgodunko@rostel.ru>
Date: Wed, 06 Jun 2007 03:57:04 −0700
Subject: ANN: QtAda 0.1.0
Newsgroups: comp.lang.ada

QtAda is an Ada 2005 language bindings to Qt 4.2. Its allow easily create powerful graphical user interface on Ada. QtAda use native thread-safe signal/slot mechanism, provide access to more than 120 Qt classes, provide Ada-aware meta object compiler, support development of custom widgets and Qt Designer's custom widget plug-ins, support loading at run-time of GUI forms from Qt Designer's UI files and so on.

New stable version 0.1.0 available at

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

Questions, comments and contribution are welcome! Please send it to QtAda users mailing list qtada-users@lists.sourceforge.net.

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

From: Vadim Godunko <vgodunko@rostel.ru>
Newsgroups: comp.lang.ada
Subject: Annonce: QtAda 0.1.1
Date: Mon, 16 Jul 2007 04:16:32 −0700

We are pleased to announce the new release of QtAda 0.1.1. This is minor bug fixes release. It includes:

- support for Qt 4.3.0
- workarounds for bugs in GNAT GPL 2007 compiler
- bug fixed in amoc compiler

[…]

Ada-spread binding

From: Per Sandberg <per.sandberg@bredband.net>
Newsgroups: comp.lang.ada
Subject: ANN: Ada-spead 1.3
Date: Fri, 24 Aug 2007 06:21:49 +0200

Ada spread 1.3

The third release of Ada-spread, an Ada 2005 binding to the performance messaging service http://www.spread.org is available at:

http://sourceforge.net/projects/ada-spread/

C2Ada

From: Nasser Abbasi <nma@12000.org>
Subject: C2Ada port to linux updated.
Date: Mon, 13 Aug 2007 03:14:42 −0700
Newsgroups: comp.lang.ada

I've just updated c2ada so that it now builds now on linux 2.6.20. The updated source code and instructions how to build are here

http://12000.org/my_notes/ada/c2ada_port/index.htm

There is example of how to run it and the ada files generated.

This tool seems useful in translating C header files.

From: Jeffrey Creem <jeff@thecreems.com>
Newsgroups: comp.lang.ada
Subject: Re: C2Ada port to linux updated.
Date: Fri, 17 Aug 2007 02:15:00 GMT

I've done some of the initial work to setup the project. I setup a SVN repository. Imported the older version, attempted to overlay your updates and setup the initial webpage for it based largely on the original html file inside the distribution.

http://c2ada.sf.net

I have not yet uploaded the zip files themselves.

If you get a SourceForge account (I'd recommend it), I'll add you as project admin to ensure you can continue to make updates in the SVN repository and/or other tasks associated with the project.

From: Nasser Abbasi <nma@12000.org>
Newsgroups: comp.lang.ada
Subject: Re: C2Ada port to linux updated.
Date: Mon, 13 Aug 2007 23:36:54 −0700

[…]

I have no idea how good one would consider the quality of the Ada code this tool generates, from adahome […] it says:

"This tool, released by Intermetrics, is based on cbind (Ada-to-C binding generator), a tool previously made public by Rational Software Corporation. C2ada is capable of generating thin Ada bindings, by translating C header files into Ada package specifications, and in addition is capable of translating C functions and statements into Ada package bodies. C2ada will do about 80% to 90% of the work of producing a thin binding or a translation, but the last 10% to 20% of the work must still be done manually. The program is free, includes source code, has no warranty, and is released to the Ada community in the hope that it will be useful. Intermetrics has used C2ada to produce Microsoft Windows, X Windows, and GCCS bindings"

[See also "Cbind" in AUJ 27-4 (Dec 2006), p.202. —su]

Konada.Db — Oracle Access Library

From: Frank Piron <frank.piron@gmail.com>
Newsgroups: comp.lang.ada
Subject: Announce: Konada.Db
Date: Mon, 02 Jul 2007 14:07:33 +0200

A new version of our Oracle Access Library Konada.Db is available at http://konad.de/download.htm

Some extensions and bugfixes e.g. for unqualified number columns in Oracle 10g are made.

[See also same topic in AUJ 25-2 (Jun 2004), pp.51–52. —su]

GTKAda contributions

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: GtkAda contributions v1.7
Newsgroups: comp.lang.ada
Organization: cbb software GmbH
Date: Thu, 5 Jul 2007 22:07:54 +0200

The software is proposed as a contribution to GtkAda, an Ada bindings to GTK+. It deals mainly with the following issues:
- Tasking support;
- Custom models for tree view widget;
- Custom cell renderers for tree view widget;
- Resource styles;
- Capturing resources of a widget;
- Embeddable images;
- Some missing subprograms and bug fixes;
- Improved hue-luminance-saturation color model;
- Simplified image buttons and buttons customizable by style properties;
- Controlled Ada types for GTK+ strong and weak references;

- Simplified means to create lists of strings.

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

The version 1.7 adds:

1. Trace procedures were added to Gtk.Main.Router to provide simple means for tracing. Trace is written in a dialog box. The dialog box can be switched between modal and modeless states to break or only record upon message written.

2. Erase was added to Gtk.Missed to remove all items of a container;

3. Get_Visible_Range was added to Gtk.Missed to determine the range of visible rows in a tree view;

4. The function Get_Background_Area was added Gtk.Missed to replace incorrect implementation of Gtk.Tree_View.

[See also same topic in AUJ 28-2 (Jun 2007), pp.75–76. —su]

Hibachi status

From: Tom's Hibachi musings
Date: Wednesday 20 June 2007
Subject: Next phase
RSS: http://hibachitom.blogspot.com/feeds/posts/default

I think the initial interest is known now. We've got participation from Ada vendors, the open source Ada community and Academic institutions. I will be presenting Hibachi at Ada Europe next week, and SigAda has agreed to a Hibachi workshop at the annual meeting in November in DC this year.

I've had some contacts and interest from industrial partners, but so far more as users than contributors.

Now we need to define each contributor's role and methods of working. Then verify/revise the project plan based on our roles and have the project review.

From: Tom Grosman <grosman@aonix.fr>
Date: Fri, 10 Aug 2007 18:53:42 +0200
Subject: Re: Schedule
Newsgroups: eclipse.tools.adt

> The original project announcement talked about an initial release of source code this month "ADT 0.5.0 — Initial release. 8/2007"

> Is that still expected?

We're not there yet. We will be having the Project Review in the middle of September. At that point, the CVS repository can be provisioned with the sources.

[See also "Hibachi — Eclipse Ada Development Tools" in AUJ 28-2 (Jun 2007) pp.81–84 and "Hibachi Events" in this issue. —su]

Aonix ADT Eclipse Plugin Installation

From: Pieter Thysebaert
Subject: Debian Etch GCC GNAT 4.1 / Aonix ADT Eclipse Plugin
Date: Mon, 16 Apr 2007 11:33:37 +0200
Organization: Ghent University
Newsgroups: comp.lang.ada

I'm on Debian GNU/Linux Etch and have installed GCC 4.1 along with the corresponding GNAT packages.

I also have Eclipse 3.2 installed and downloaded (through Eclipse Software Updates) the Aonix ADT plugin.

My problem is that the ADT does not pick up Etch's GCC GNAT 4.1 toolchain (when selecting GNAT Linux Toolchain), no matter what directory I feed it (I naively thought that setting the toolchain path to /usr/bin would do the trick).

Is there anyone who knows how to make the Aonix ADT plugin work together with the GCC 4.1 based GNAT from Debian Etch?

Or is this a known limitation of the plugin (i.e. that it only works with AdaCore GNAT Pro and GNAT GPL) ?

From: Pieter Thysebaert
Subject: SOLVED: Debian Etch GCC GNAT 4.1 / Aonix ADT Eclipse Plugin
Date: Tue, 17 Apr 2007 09:25:36 +0200
Organization: Ghent University
Newsgroups: comp.lang.ada

[…] It turns out that specifying "/usr" as the toolchain path (instead of /usr/bin) does the trick… (i.e. makesd Ao Aonix ADT pick up and recognize the GNAT toolchain).

From: Tom Grosman <grosman@aonix.fr>
Newsgroups: comp.lang.ada
Subject: Re: SOLVED: Debian Etch GCC GNAT 4.1 / Aonix ADT Eclipse Plugin
Date: Tue, 17 Apr 2007 17:09:44 +0200
Organization: Aonix

> Actually that is a good tip and makes sense. The way the plugin works, it needs to find more than just the compiler and tools — it needs libraries, specs, etc.

[…] In order for AonixADT to recognize a GNAT toolchain, The "toolchain directory" that is specified should contain the GNAT bin subdirectory (GNAT executables are located here) to which you should have execution permission. AonixADT then checks within the GNAT bin directory for the presence of the executable file "gnatmake" to determine if you have specified a valid toolchain location. So far, this seems to work no matter the O/S nor compilation environment (gnuada, GNATPRO).

From: Tom Grosman <grosman@aonix.fr>
Newsgroups: comp.lang.ada
Subject: Re: SOLVED: Debian Etch GCC GNAT 4.1 / Aonix ADT Eclipse Plugin
Date: Tue, 17 Apr 2007 22:29:14 +0200
Organization: Aonix

> Won't work for cross-compilers that have a prefix (eg. powerpc-wrs-vxworks-gnatmake). So, you might want to search for filenames containing the "gnatmake" substring if you don't already (though I suspect you may in order to handle the Windows case).

Thanks for the info, Ed. We (Aonix) haven't integrated any of the GNAT cross compiler toolchains. I reckon that it will be done in the context of Hibachi, and not AonixADT. And the underlying techno of Hibachi is open to change as the project evolves to be as open and extensible as possible.

Ada-related Products

AdaCore — GNAT Pro for DO-178B

From: AdaCore Press Center
Date: Tuesday June 19, 2007
Subject: AdaCore Announces the GNAT Pro High-Integrity Edition for DO-178B
RSS: www.adacore.com/category/press-center/feed/

Specialized development environment and tool set supporting safety-critical standards

NEW YORK and TAMPA, Fla., June 19, 2007 — Systems & Software Technology Conference — AdaCore, provider of the highest-quality Ada tools and support services, today announced the immediate availability of the GNAT Pro High-Integrity Edition for DO-178B. The product is a specialized version of GNAT Pro that provides an independently certified development environment and tool set specifically targeted towards developers needing to satisfy this demanding safety-critical standard. It is the first in a series of specialized High-Integrity Edition packages designed to provide complete support for specific safety or security industry requirements, such as RTCA DO-178B / EUROCAE ED-12B and other high-integrity standards. Additional High-Integrity Edition packages will be announced as they become available.

"AdaCore recognizes that our customers require off-the-shelf solutions to meet high-reliability and safety-critical software development standards," said Robert Dewar, President of AdaCore. "Customers are currently using GNAT Pro to develop avionics systems that need to satisfy the DO-178B Level A standard, and our run-time library has been certified to meet these requirements. The GNAT Pro High-Integrity Edition for DO-178B provides the associated life-cycle artifacts along with the development tools

necessary to comply with this and similar safety-critical standards."

AdaCore has also taken the additional step of having an independent organization perform the full certification of the Ada run-time library that is available for GNAT Pro High-Integrity Edition for DO-178B. Verocel, a company with recognized expertise in this domain, has developed the certification package.

"AdaCore has a consistent policy of using world-class experts to prepare certification materials to ensure that certification is 100% independent and not influenced by the development team," stated Robert Dewar. "The effectiveness of DO-178B relies on an impartial certification process, and we strongly feel that this is best assured by having highly qualified third parties prepare the certification material and perform the required DER (FAA Designated Engineering Representative) review."

About GNAT Pro High-Integrity Edition for DO-178B

GNAT Pro High-Integrity Edition for DO-178B is an enhanced version of the GNAT Pro technology, designed for building safe and secure software. Formerly known as GNAT Pro HIE, its many features help to reduce the cost of developing and certifying systems that have to meet safety standards such as RTCA DO-178B / EUROCAE ED-12B and other high-integrity standards. The package includes a full, multi-language cross-compile system, a configurable Ada run-time system, and integration with best-in-class test capabilities. The run-time library for the GNAT Pro High-Integrity Edition for DO-178B has already been certified to the highest safety level for DO-178B Level A, as a part of multiple avionics systems. These life cycle artifacts are available with the package. Further details on this new product package can be found at: http://www.adacore.com/home/gnatpro/safety-critical

Availability

GNAT Pro High-Integrity Edition for DO-178B is immediately available 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 software solutions for Ada, a modern programming language designed for large, long-lived applications where reliability, efficiency and safety are critical. AdaCore's flagship product is GNAT Pro, which comes with expert online support and is available on more platforms than any other Ada technology. AdaCore has customers worldwide; see http://www.adacore.com/home/company/customers for more information.

Use of Ada and GNAT Pro continues to grow in high-integrity and safety-critical applications, including avionics, defense, air traffic control, railroad systems, financial services and medical devices. AdaCore has North American headquarters in New York and European headquarters in Paris. www.adacore.com

[See also "AdaCore — GNAT Pro 6.0.1" in AUJ 28-1 (Mar 2007), pp.11–12. —su]

AdaCore — Support for .NET and Vista

From: AdaCore Developer Center
Date: Friday July 27, 2007
Subject: Support for .NET and Vista
RSS: www.adacore.com/category/developers-center/development-log/feed/

As part of the 6.0.2 release, AdaCore is pleased to announce that support for the Windows OS has been extended to the new Windows Vista platform. GNAT Pro now supports all Windows platforms from Windows 2000 through to Vista.

In the near future AdaCore will announce support for Microsoft .NET bringing a commercial Ada development environment to this platform for the first time. The port will include support for Microsoft Visual Studio .NET 2005.

A press release providing more details on these ports will be issued in the coming months.

Adalog — AdaControl

From: Jean-Pierre Rosen <rosen@adalog.fr>
Newsgroups: comp.lang.ada
Subject: AdaControl 1.7 released
Date: Thu, 05 Jul 2007 17:29:24 +0200
Organization: Adalog

We are happy to announce the availability of version 1.7 of AdaControl, the free Ada controlling tool.

This version features 289 possible checks, not counting parametrizations!

In addition, there is even better integration with GPS (works with GPS4.x now), and AdaGide (thanks Gautier de Montmollin).

Please visit http://www.adalog.fr/adacontrol2.htm for a detailed description of old and new features.

[See also same topic in AUJ 27-4 (Dec 2006), p.206. —su]

Aonix — Ameos UML goes open source

Aonix Contributes Ameos UML Technology to Open Source
AMEOS Model Driven Architecture Tools Made Available Under LGPL
Paris, France, June 20, 2007

Aonix®, a provider of solutions for mission-critical applications development, announced today that it is contributing its powerful Ameos modeling technology to the open source community. Ameos, based on the pioneering Software through Pictures modeling tool family, offers UML profiles to generate C/C++, Ada, Java, CORBA, COM, and EJB. Under the new open source policy, Ameos is available under terms based on the GNU Lesser General Public License (LGPL) as OpenAmeos. This open source strategy for Ameos allows Aonix to focus marketing resources on its expanding PERC technologies for real-time Java developers, while providing the tools to ensure long-term value to Ameos users.

Ameos implements UML 2.0 profiles, model-driven architecture (MDA) -based model transformation and a modern, convenient user interface—features designed to powerfully meet the modeling needs of modern and complex software systems. Through its UML profiles, developers can easily extend standard UML notation and adapt it to project-specific needs. Since the transformation engine is based on MDA architecture, design time is reduced as the model process is able to mature to a greater level prior to implementing target-specific detail.

"Ameos is a powerful and stable technology," said Dave Wood, Aonix VP marketing. "We believe that the best means of expanding its adoption and evolution comes by donating the source code to Ameos users. It's always exciting to see how open-source communities generously make their own contributions available to others."

For OpenAmeos, strong community support is already in place. ScopeSET, a leading expert in Ameos technology, has partnered with Aonix in this open source initiative. ScopeSET will provide product support and professional services for Ameos and customer-requested derivatives.

"With a long history of developing and supporting Ameos and MDA tools, ScopeSET is pleased to lend its expertise to this important Aonix initiative," said Armin Mueller of ScopeSET. "Our team is committed to continue providing extensive tool-specific know-how to the OpenAmeos community. We will also act as integrators for future OpenAmeos distributions to ensure quality and further development."

Ameos is the second major technology contributed by Aonix to the open source community this year. The decision illustrates Aonix' commitment to selecting licensing terms that best suit the needs of users of each of its product lines. In April, Aonix announced the open

source availability of the Aonix Eclipse Ada Development Toolkit. Aonix is leading the new Eclipse-based initiative to create an Ada Development Toolkit (ADT) project and will collaborate with the Eclipse Foundation™ toward that end.

Under the terms of the Ameos open source agreement, a "clean" open source version of Ameos, called OpenAmeos, has been created to ensure that anyone installing a new version of Ameos will be able to use it freely without encumbrances of any other source contributions. This version is freely downloadable at http://www.openameos.org. In addition, any company or academic institution who wants to distribute OpenAmeos source code is required to also make their changes to the source code freely available to others in order to ensure the continuing open evolution of Ameos.

About Aonix

Aonix offers mission- and safety-critical solutions primarily to the military and aerospace, telecommunications and transportation industries. Aonix delivers the leading high-reliability, real-time embedded virtual machine solution for running Java™ programs deployed today and has the largest number of certified Ada applications at the highest level of criticality. Headquartered in San Diego, CA and Paris, France, Aonix operates sales offices throughout North America and Europe in addition to offering a network of international distributors. For more information, visit www.aonix.com.

[See also "Aonix — AonixADT goes open source" in AUJ 28-2 (Jun 2007), p.81. —su]

From: Martin <martin.dowie@btopenworld.com>
Newsgroups: comp.lang.ada
Subject: New open source UML tool including Ada support
Date: Tue, 10 Jul 2007 02:38:38 −0700

Don't know why Aonix haven't bothered to post here but they've opened up their UML tool "Ameos" as "OpenAmeos"

Haven't spent more than 5 minutes looking at it yet but it seems to offer both forward source code generation and reverse into UML support for both Ada95 and what looks like support for "Ravenscar" Ada 95.

Newsgroups: comp.lang.ada
Subject: Re: New open source UML tool including Ada support
From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Tue, 10 Jul 2007 20:28:37 +0200

> I also fear (since the tool you get is an exe file) that it won't be easily portable to Unix (but this is another problem).

I think Ameos has a few ideas in common with Software through Pictures™, a venerable tool. "Since 1985 we have been developing StP consistently and adapting our products to the latest technologies and requirements. Today structured methods (StP/SE) like Structure Analysis and Structured Design are supported, as well as the Unified Modeling Language (StP/UML) in the OO range."

There are a few snapshots showing Motif interfaces.

From: Dave Wood <dave.wood@aonix.com>
Newsgroups: comp.lang.ada
Subject: Re: New open source UML tool including Ada support
Date: Mon, 16 Jul 2007 17:38:26 −0700

> […] But at the moment there is only an executable, no source. And I have been wondering about the license this executable is under, since anybody getting it, can't redistribute it under the LGPL/modified-whatever, because, well he hasn't gotten the source (and probably not not LGPL license too, because that would guarantee him the source).

[…]

We decided to put Ameos into open source, and we made certain sources available IMMEDIATELY to existing Ameos users on an as-needed basis. As such "is available" is not inaccurate. We also went ahead and announced the open sourcing so other people would know about it. I don't want to debate press release semantics, but IMO this one is fine.

Why the delay? The reality is that a couple components of the sources need to be "cleansed" of encumbrances before the sources can be put on the web site for general availability. To be distinguished, this cleansed version is called OpenAmeos.

This should happen within a few weeks. Unless you happened to have a burning need for Ameos sources RIGHT NOW, I wouldn't expect this interim period to be too painful for anybody. As you can see, the executable is provided for anyone with actual work needing to be done, and is indeed free as beer and will make you almost as happy as drinking free beer. […]

The OpenAmeos sources and LGPL license agreement will appear "soon" on the OpenAmeos.org web site. Please enjoy your free software, and free sources, which follow many, many years of hard work by many talented engineers.

Regards,

Dave Wood
VP Marketing
Aonix

Newsgroups: comp.lang.ada
Subject: Re: New open source UML tool including Ada support
From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Tue, 17 Jul 2007 11:36:54 +0200

Indeed, the software has made me see the tools again that I once had an opportunity to evaluate (as StP/UML). I'd like to point out that Ameos has a number of features not usually present in UML tools. Among these are linking parts of specification documents to parts of the model and performing model checks.

IIRC, code generation is/was based on a template mechanism (these were updated/reworked/… some time ago).

FWIW, the license text can be read now by performing an installation of the software.

Aonix — ObjectAda for Windows V8.4

Aonix ObjectAda Supports Windows Vista, .NET 2005
Best selling Ada technology updated for latest Microsoft platform
San Diego, August 27, 2007

Aonix®, a provider of solutions for safety- and mission-critical applications, announced the release of ObjectAda for Windows V8.4. ObjectAda for Windows, the most popular commercial Ada development solution for Windows platforms, provides a complete enterprise-level environment for the development of Windows applications using the Ada programming language. This latest release now enables development on the Microsoft Windows Vista platform and lets developers use ObjectAda with the Microsoft Visual Studio .NET 2005 tools. ObjectAda for Windows also plugs seamlessly into the Eclipse environment.

ObjectAda for Windows 8.4 includes the comprehensive Ada libraries needed for calling Windows Win32 and the Visual C++ .NET 2005 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 2005 tools and libraries. ObjectAda for Windows can either be used standalone or in combination with the Visual Studio .NET 2005 compilers and the latest Windows Platform SDK. ObjectAda for Windows generates symbolic debugging information compatible with the Visual Studio .NET 2005 debugger and thereby enables debugging of complex applications written in multiple languages, such as C/C++ and Ada.

"ObjectAda for Windows v8.4 builds on the strengths of this product evident from its inception," noted Dave Wood, vice president of marketing at Aonix. "Its strengths are an easy-to-use development environment, excellent compiler performance, and capacity to support development of large and complex mission-critical applications. This latest release enables customers to use

ObjectAda in combination with the current suite of Visual Studio .NET 2005 compilers and tools from Microsoft and use the Windows Vista platform for the long-term evolution and maintenance of mission-critical applications."

In addition to the basic compiler development package, an upgrade package called ObjectAda Project Pack contains AdaJNI, an interface to call Java™ programs from Ada, and the AdaNav™ toolset, which provides complete system HTML source-navigation capabilities as well as call- and unit-tree graphical reporting and automatic data dictionary generation. The AdaNav profiler also offers run-time performance reporting to identify application hot spots.

ObjectAda for Windows gives developers the choice between the traditional Aonix IDE for development and the new AonixADT™ Eclipse plug-in. Geared to maximize developer ease and efficiency, 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.

[See also "Aonix — ObjectAda for Windows 8.2" in AUJ 27-4 (Jun 2007), p.207. —su]

Excel Software — WinA&D and WinTranslator

WinA&D and WinTranslator for PHP

Software Design, Requirements Management and Model Generation Tools

August 10, 2007 — Excel Software is shipping a new version of the WinA&D modeling tool and WinTranslator reengineering tool that supports PHP software design, code generation and model generation from code.

PHP is a modern scripting language for procedural and object-oriented programming. It drives popular web sites that often grow into large-scale development projects. WinA&D adds PHP modeling enhancements and code generation from class diagrams or structure charts. Often PHP programs provide the glue between web pages and a database engine. Data models drawn in WinA&D can generate the SQL schema.

WinTranslator scans PHP source code to extract design information presented as class diagrams and structure charts in WinA&D. Structural diagrams and dictionary information extracted from legacy source code, class frameworks or open source projects yield reusable code assets in minutes. WinA&D and WinTranslator provide a rich, scalable modeling environment with code generation from models and model generation from code.

- UML Class Models to and from C++, C#, Java, Delphi, Ada, or PHP
- Rich Data Models to and from SQL
- Structure Charts to and from C, Pascal, Delphi, Basic, Fortran or PHP

WinA&D is a comprehensive tool for system models and simulation, requirements management, structured analysis and design, object-oriented UML, multi-task and database design. The new release streamlines requirements traceability, adds fast global search across thousands of files and folders and fully automates model generation of multi-threaded software systems.

PHP programs are event driven by end-user actions like clicking a button or selecting a menu on a web page. A user event may trigger a thread of execution running thousands of lines of PHP code. WinTranslator and WinA&D sift through and identify execution threads, then present each in a structure chart. Reused program branches are presented on a shared diagram. The same automated process applied to embedded, real-time C code requires little human effort to reveal the structure of a large software system.

The new WinA&D 5.1.1 and WinTranslator 3.0.1 release is a free download for current 5.1 and 3.0 customers. WinA&D for Windows has a Standard edition at $495, Desktop edition at $1195 and Developer edition at $1995. A site license allows multi-user, team dictionary and requirements. WinTranslator is $495. See www.excelsoftware.com for product information, demo editions, pricing and secure online ordering.

Company Contacts

Excel Software
Ph: (505) 771-3719
Fax: (505) 771-3718
Email: info@excelsoftware.com
Web: http://www.excelsoftware.com

McKae Technologies — Avatox 1.8

From: Marc A. Criley <mc@mckae.com>
Date: Fri, 31 Aug 2007 20:10:36 −0500
Newsgroups: comp.lang.ada
Subject: Announce: Avatox 1.8 is now available

Avatox (Ada, Via Asis, To Xml) is an application that traverses Ada compilation units and outputs their ASIS representation(s) as XML document(s) in the Avatox XML Format, AXF, file extension "axf". The format of the XML in the document can be configured, and supplemental source annotations can be generated.

Changes since version 1.7:

- Fixed a problem with private parts of various kinds of units not being recognized as private.
- Fixed a letter casing problem with rep specs.
- Numerous upgrades to the AXF2Ada stylesheet.

In this release is the in-work version of axf2ada.xsl, an XSLT stylesheet that converts AXF into Ada source code. It is a work in progress, and currently can regenerate the Avatox and DTraq dtqserver source code from their AXF representations. For more information about axf2ada, see the Avatox web page.

Avatox 1.8 is available at www.mckae.com/avatox.html.

Date: Sat, 01 Sep 2007 07:43:16 −0500
From: Marc A. Criley <mc@mckae.com>
Newsgroups: comp.lang.ada
Subject: Re: Announce: Avatox 1.8 is now available

> Interesting — I still wonder why not XMI. With XMI we could feed the output into tools like umbrello [2] which would be the first step towards an Ada UML tool with round trip engineering.

That's a fair question.

AXF is seen as a first step. First let's get the Ada into a more malleable form that can be processed by XML tools and technologies. Then it becomes more amenable to content extract and transformation, with Avatox generating "AXFPoint" (AXF Points Of INformation for Transformation) elements to to assist with transformations by providing information beyond that of the basic ASIS-derived semantic information.

In other words, AXF -> UMI would be easier than Ada -> UMI because you've lessened the "impedance mismatch" between the source and target. In addition, if you want to change the way Ada constructs are mapped to UMI (or any other target), or if you need to adapt to a revised UMI, it's almost always going to be much simpler to alter an XML processor—especially if it's stylesheet based—than the Ada-to-XML generator.

Date: Sat, 01 Sep 2007 15:33:46 −0500
From: Marc A. Criley <mc@mckae.com>
Newsgroups: comp.lang.ada
Subject: Re: Announce: Avatox 1.8 is now available

> So:

> .adb ==Avatox==> .axf ==XXX==> .xmi ==xmi2code==> .adb

> is easier than

> .adb ==Avatox==> .xmi ==xmi2code=> .adb

Actually, yes :-)

The reason is that adb => AXF is an almost mechanical translation, AXF being an XML representation of the ASIS-

derived structure of the source program. And not least, Avatox is pretty much done.

Going directly from Ada to XMI, however, is a much more analytical problem, requiring the mapping of Ada constructs to XMI equivalents. And from what I've experienced with Source Code 2 Design Model converters (for any language) they don't do a great job. I mean, what comes out is a pretty prosaic representation of the _code_ as UML (or whatever) artifacts. You would not have diagrammed the system this way, and to try to employ some higher level intelligence in the converter to recognize programming idioms and abstractions and then map _those_ to "better" XMI requires a lot more work (and the fact that I don't see it in commercial tools tells me that it requires a LOT more work). Also, AXF retains _all_ the information in the original source, including comments, line/column extents (which gives you the original whitespace), etc. Going from source code to model you tend to start omitting things because of the differing levels of abstraction.

> Only we would need XXX

Yes, which could be built incrementally. Instead of trying to go straight from AXF (or Ada source) to XMI, you can do some filtering, some simple transformations, some more filtering, some pattern recognition, some transformations, and eventually end up with XMI.

In this case you then have:

.adb ==[Avatox+AXF2XMI]==> .xmi ==xmi2code=> .adb

And there you go! Have at it! :-)

[See also "McKae Technologies — Avatox 1.7" in AUJ 28-2 (Jun 2007), p.84. —su]

Objektum — OOUnit

Objektum Launches ALM Toolset Plug-In Programme

Croydon, UK – July 4th, 2007 – Objektum (www.objektum.com), the leading European provider of software training and consulting services, today announced that it has launched a programme to provide support for Application Lifecycle Management (ALM) toolsets. Objektum has developed two plug-ins for the acclaimed ARTiSAN Studio® toolset:

Karner Use Case Based Estimator

Use case modelling is a technique that has been widely used throughout the software industry to describe and capture the functional requirements of a software system. Determining use case points is now becoming the de-facto method for estimating software development early in the lifecycle. Objektum's use case based estimator plug-in utilises the Karner method to determine the effort/cost required to fully implement a use case, or collection of use cases.

Through this fully-featured plug-in, ARTiSAN Studio® users can now estimate the effort and cost associated with developing a use case, or set of use cases that form a release, held in the model. Technical and environmental factors as well as cost weightings can be adjusted and saved within the model allowing project managers to more accurately manage software releases.

OOUnit

The testing of object-oriented software is fundamentally different from the testing of non-object-oriented software due to such factors as Information hiding, Encapsulation and Inheritance. The OOUnit framework integrates with ARTiSAN Studio® to overcome these challenges to provide a seamless environment for developers and testers to generate automated test cases, build upon the best practise of well-known test tools, such as IPL's AdaTEST and the AUnit framework.

By simply right-clicking a class in ARTiSAN Studio®, the test infrastructure can be automatically generated (attributes, operations, parameters etc), taking the tedium out of unit, integration and system testing. Stubs can be automatically created allowing testers to concentrate on the test sets (initial data and expected results) and not the labour intensive frameworks or scripts normally associated with testing. Currently OOUnit supports the Ada programming language but a C/C++ variant is to be launched soon.

Derek Russell said "At Objektum we have used our insight into the UML development lifecycle and toolset technologies to launch an ALM toolset plug-in programme." He adds "We see our plug-ins as enabling the convergence of technologies to maximise the efficiency of managers, analysts, developers and testers. We are working closely with companies such as ARTiSAN Software to develop a suite of fully integrated tools that support the entire software development lifecycle."

About Objektum

Objektum is an ISO 9001:2000 certified, independent provider of tailored training, consultancy and mentoring, specialising in object oriented, real-time and embedded systems. Its customised courses and workshops are designed to provide the skills, tools and knowledge necessary to implement systems accurately and de-risk projects.

Objektum's consultancy and mentoring service provides its technical experts to work alongside existing teams assisting in the integration of new technologies and processes. Objektum is a specialist in the UML2.0, SysML and CORBA standards.

Objektum's approach is fresh and innovative. Its unique training techniques add value by enhancing the effectiveness of its courses and workshops whilst its technical expertise offers creative solutions to industry problems.

Objektum participates in industry organisations such as OMG, INCOSE, Ada-UK and others to maintain its cutting edge position within the systems and software development arena – Objektum also provides software development services.

Objektum features several leading FTSE 100 companies in its client list.

Contacts:
Andy Bissell, Derek Russell
Objektum Ltd.
Phone: 0800 019 49 50
E-mail: info@objektum.com

Ada and CORBA

OMG Data Distribution Service bindings

From: Per Sandberg <per.sandberg@bredband.net>
Newsgroups: comp.lang.ada
Subject: DDS and Ada
Date: Fri, 22 Jun 2007 14:54:33 +0200

We are in the process of producing a thick Ada 2005 binding to DDS the current plan is to write a binding that resembles of the Java binding specified by OMG. And the binding shall only depend on the C libraries for DDS as specified by OMG.

Artifacts to be produced:

- IDL to Ada generator
- Support packages for the generated code.
- Bindings as specified by the DDS.idl file.

The current path is to use the idl2Ada specification for CORBA and and then tweak the bindings to resemble of the DDS Java interface.

AdaCore — OMG tags assigned to PolyORB

From: AdaCore Developer Center
Date: Thursday August 2, 2007
Subject: [polyorb] Official OMG tags assigned to PolyORB
RSS: www.adacore.com/category/developers-center/development-log/feed/

PolyORB has received official OMG profile tags, service tags, component IDs, vendor minor code IDs and ORB type IDs from the OMG. Note that users who rely on PolyORB-specific features across partitions in an application (including all users of the DSA application personality) will have to upgrade all partitions at the same time so that they use a consistent set of tags.

Ada and GNU/Linux

Debian transition to GCC 4.2

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Newsgroups: comp.lang.ada
Subject: Ada in Debian: transition to GCC 4.2
Date: Thu, 28 Jun 2007 19:19:12 +0200

I've just completed the task of updating the existing patches from GCC 4.1 to GCC 4.2. These patches introduce libgnatvsn and libgnatprj, and link gnatmake and friends dynamically rather than statically. Now, the next task I've assigned myself is to build the setjump/longjump version of libgnat in addition to the zero-cost exception mechanism. Would anyone be interested in participating?

I am planning to upload gnat-4.2 without SJLJ this weekend or early next week, after today's upload of gcc-4.2 has been built on all architectures. I have not yet started work on SJLJ; that will come in a future upload. Help is appreciated to test GCC 4.2, build the SJLJ version of the library, and move gnat-glade (the distributed systems) to GCC 4.2 with SJLJ.

If you want to build gnat-4.2 for yourself, go to [1] and read the files README.maintainers and TODO.

GCC 4.2 is already the default Fortran compiler in Debian unstable. The plan is to transition C and C++ packages to GCC 4.2 in the next weeks [2], with the ultimate goal to use GCC 4.2 as the default compiler for all languages except Java.

It is not yet known at this point whether GCC 4.3 will be released in time for Lenny, the next stable version of Debian which is scheduled for release in October 2008 (the toolchain freeze will take place 6 months prior to that). So, I would like to complete a transition of all Ada packages to 4.2 before considering 4.3. That transition requires that all items in the TODO list be completed first.

Ada and Macintosh

GPS 4.0 for Mac/PowerPC

Date: Tue, 05 Jun 2007 13:20:44 +0300
Subject: Re: Looking for GPS 4.0 for Mac/PowerPC
From: Martin Krischik <krischik@users.sourceforge.net>
Newsgroups: comp.lang.ada

> unfortunately the latest GPL-ed version of GNAT/GPS (2007) is available only for Windows and Linux, the Mac/PPC binary is no longer offered by AdaCore. I am wondering whether it will be made available soon, or, if not, is there anyone who compiled it for the Mac/PPC? I tried to compile the system from the Linux sources but there are so many libraries missing that I gave up… :-(

All the libraries needed are available from The GNU Ada Project. If you search our repository you will also out which lib to compile first.

Now the bad news: GPL 2007 was not able to compile the newest GPS 4.1.1 — in fact no available compiler is. The next best — GPS 4.1.0 — can be compiled with GCC 4.2.0 but again not with GPL 2007. Probably the reason why GNAT/GPL 2007 comes with a quite outdated GPS.

[1] http://gnuada.sf.net

From: Simon Wright <simon.j.wright@mac.com>
Newsgroups: comp.lang.ada
Subject: Re: Looking for GPS 4.0 for Mac/PowerPC
Date: Sun, 03 Jun 2007 18:55:12 +0100

I believe there was some problem building gnat-gpl-2007 … but there is an FSF compiler here:

http://www.macada.org/macada/Welcome.html

References to Publications

ARTiSAN Newswire

[Extracts from the table of contents. See elsewhere in this news section for selected items. —su]

Date: 11 Jul 2007 14:28:28 −0000
To: santiago.uruena@upm.es
From: ARTiSAN Software Tools <info@artisansw.com>
Subject: ARTiSAN Newswire — July2007

Welcome to the July edition of the ARTiSAN Newswire. In this issue we bring you news about our NEW SysML poster, Objecktum Plug-In programme, Events we are attending and Training course Information.

Latest News

Objektum Launches ALM Toolset Plug-In Programme

As part of Objektum's Application Lifecycle Management (ALM) Plug-ins, Objektum now offers the following plug-ins for the acclaimed ARTiSAN Studio® toolset:

- Karner Use Case Based Estimation plug-in: A fully integrated plug-in that allows project effort and cost to be estimated based on use cases held in the model.

- OOUnit: An integrated plug-in that generates executable test cases from classes held in the model. The test framework exploits the features of AdaTest or AUnit (C / C++ coming soon).

[…]

Upcoming events for your diary:

The Ada UK Conference 2007. Sept 25th – Manchester, UK

This event is organised to promote awareness of the Ada programming language, and to highlight the increased relevance of Ada in safety- and security-critical programming. We will be exhibiting and presenting a paper on "Implementing Design Patterns in Ada".

Register now: http://www.ada-uk-conference.co.uk/index.html

[…]

To find out more about this course and the other courses we offer, please visit our Service and Solutions are on our website:

http://www.artisansw.com/services/training/courses/

EE Times

From: Bob Spooner <rls19@psu.edu>
Newsgroups: comp.lang.ada
Subject: reply to article in EE Times
Date: Mon, 9 Jul 2007 16:10:15 −0400
Organization: Penn State University, Center for Academic Computing

Robert Dewar, President and CEO of AdaCore has replied in the Crosstalk column of the July 2, 2007 issue of EE Times to some statements in Richard Goering's article "Ada 2005 speaks to real-time embedded applications." Look on page 48.

Application Software Developer

From: AdaCore Developer Center
Date: Wednesday July 11, 2007
Subject: Need Secure Software?
RSS: www.adacore.com/category/developers-center/development-log/feed/

Following on from Ben Brosgol's SSTC paper, this week sees the publication of an interesting article on language choice for security-critical software:

Security in modern embedded systems is critical — software developers must keep ahead of the "bad guys". A key decision is the choice of programming language: while some languages make it easier to produce secure code, others seem to exacerbate rather than solve the problem. This article identifies the key requirements that a programming language must satisfy, and shows how Ada effectively meets these demands.

To read the article in full, [see http://www.applicationsoftwaredeveloper.com/features/june07/article2.html —su]

GNAT Pro Insider newsletter

From: AdaCore Developer Center
Date: Wednesday July 4, 2007
Subject: Spring newsletter available
RSS: www.adacore.com/category/developers-center/development-log/feed/

The latest edition of the GNAT Pro Insider newsletter is now available. This edition includes articles on:

- What's New in GNATbench 2.0
- Major New Air Traffic Control System Using GNAT Pro
- Current Releases
- Spotlighting a GAP Member
- Ada 2005 is an Official ISO Standard!
- In the Pipeline
- Interview with Arnaud Charlet
- Technology Webinars
- AdaCore Partner Vector Software Helps Certification Effort for DO-178B
- AdaCore at Conferences

To download the newsletter [see http://www.adacore.com/wp-content/files/attachments/adacore_news_0607_web.pdf —su]

SSTC

From: AdaCore Developer Center
Date: Monday July 2, 2007
Subject: Designing High-Security Systems
RSS: www.adacore.com/category/developers-center/development-log/feed/

Ben Brosgol's presentation from the recent SSTC event entitled "Designing High-Security Systems: A Comparison of Programming Languages"

The high degree of interconnectivity in today's computing systems and the increasing threat from technically sophisticated adversaries make security an essential requirement in modern military software. Many technical factors affect the ease or difficulty of meeting this requirement, including the programming language, the software development tools, the operating system, and the application program interface. This presentation focuses on the programming language, which is arguably the factor that a development project manager can control most directly, and assesses three major language families with respect to the criteria that a secure system must meet:

- Ada 2005 and the Ada-based SPARK language
- C and C++
- Java and its relevant extensions (Real-Time Specification for Java, Safety-Critical Real-Time Java)

The presentation focuses in particular on how modern language features (such as the data type model, Object-Oriented Programming ("OOP"), exception handling, and concurrency) affect application security, and compares the requirememts for security and for safety. [see http://www.adacore.com/wp-content/files/attachments/BrosgolGiccaPresentation-SSTC2007.pdf —su]

Micro Technology Europe

From: AdaCore Developer Center
Date: Monday September 3, 2007
Subject: Ada: New features, same performance for the embedded market
RSS: www.adacore.com/category/developers-center/development-log/feed/

Jose Ruiz describes some of the the advantages of using Ada for developing embedded systems in this month's Micro Technology Europe magazine.

"The complexity of embedded software increases at least at the same pace as the processing power of the processors used. Additionally, many embedded applications require high reliability or are safety-critical. To meet these demands a programming language needs high-level features that support sound software engineering…"

Avionics Magazine

From: AdaCore Press Center
Date: Thursday September 6, 2007
Subject: C-130 Avionics Modernization Program
RSS: www.adacore.com/category/press-center/feed/

Avionics Magazine

Having survived ethical, technology and cost challenges, the C-130 Avionics Modernization Program is nearing key development and production milestones

COTS Journal

From: AdaCore Press Center
Date: Thursday September 6, 2007
Subject: Ada a Winner for High-Integrity Real-Time Apps
RSS: www.adacore.com/category/press-center/feed/

COTS Journal

Despite some challenges from C++ and Java, Ada is still the technology to beat in high-integrity real-time military applications. New features in Ada 2005 help sweeten the deal.

Blue GNU — AdaCore Interview

Author: Don Parris
From: AdaCore Developer Center
Date: Wed, 2007-08-15 23:21
Subject: Ada Core Technologies: Free Software Business Model Is Viable
RSS: http://blue-gnu.biz/blog/feed

It has been said there is no such thing as a 'true' Free Software business. Blue GNU interviewed the Ada Core Technologies team to learn about the company that has been a 'true' Free Software business for over 20 years. Ada Core is one of a few businesses listed as such by the Free Software Foundation/GNU Project.

Note: In studying the company's website, I mistakenly thought that Gnat Pro must be non-Free software, but the ACT team's response helps to clarify the issues for our readers.

First off, What is Ada, and in what kinds of applications is it most commonly used? (For non-programmers in the audience)

Ada is a modern programming language that was first created in 1983 and was recently ratified by ISO for its third revision. It is primarily used in large, long-lived applications where reliability, efficiency and safety are critical, such as commercial and defense aircraft avionics, air traffic control, railroad systems, financial services and medical devices. To learn more about the language peruse the Ada Information Clearing House at: http://www.adaic.org

Can you tell me a little bit about the Executive team's involvement in the development of Ada?

Several members of AdaCore's executive staff were involved with the original GNAT Ada development effort at NYU. See:

http://www.adacore.com/home/company/exec_team.

What events led up to the launch of AdaCore Technologies?

In 1994, at the completion of the GNAT Ada project at NYU, members of the GNAT Ada development team founded AdaCore Technologies.

AdaCore was the first company to launch full language support for Ada 95, the second ISO revision of the language. AdaCore developed a full Ada compile system to support the language based on the GNU technology, called GNAT. This technology was and is still made available to the GNU community free of charge. What AdaCore provides its customers is a production quality version of this product named GNAT Pro along with top notch support for the product in terms of bug fixes and enhancement requests, as well as the industry's foremost Ada language expertise to help customers better understand the language and design their applications.

How big, in terms of employees, is AdaCore?

AdaCore has approximately 55 employees worldwide. AdaCore has North American headquarters in New York and European headquarters in Paris.

The GNU Project lists AdaCore Technologies as developing Free Software exclusively. Is that still true, or have things changed over the years? I ask because the website refers to "commercial open" technology, yet

GNAT Pro pricing appears to be on a "per seat" basis. Is GNAT Pro Free Software?

GNAT Pro is free software. However, it is important to understand that the "Free" in Free Software has nothing to do with cost, it is about the freedom of the license. Also, "commercial" is not a synonym for "non-free." That confuses two entirely different issues. A program is commercial if it is developed as a business activity. A commercial program can be free or non-free, depending on its license. The two questions, what sort of entity developed the program and what freedom its users have, are independent.

AdaCore is the original creator of GNAT and the primary maintainer of the product. Some versions are available free of charge. However, customers typically want top quality support beyond just free software. AdaCore provides this top level support via a supported version of the GNAT product named GNAT Pro along with an industry-leading staff of Ada language experts. This all adds up to our customers being successful in their development efforts. […]

The million-dollar question is, how does a business survive — never mind thrive — "selling" Free Software? Aren't you giving away the farm?

Selling Free Software is a perfectly viable business model. AdaCore has been growing at a rate of approximately 30% per year. This is due mostly to word-of-mouth about the incredible added value we provide behind the GNAT Pro product, in terms of correcting problems, adding features and such, as well as the expertise we provide to help our customers successfully use the Ada language for their development projects.

How does your model compare to that of, say, MySQL, a company that dual-licenses its software?

We use dual licensing in the context of our academic program, but for our commercial customers we use only Free Software licensing.

What is unique to Ada Core's situation that makes Free Software development a viable business model?

Free Software is a viable model in many cases. AdaCore's model is not unique. Just like Microsoft, we sell copyrighted software with support and a license allowing limited copying. The only difference is that the license gives much greater freedom to the recipient, which is a commercial advantage for our customers. Obviously, our model is also heavily support-based.

What trends do you see, away from or towards, a greater focus on Free Software businesses?

There is a clear trend towards greater use of Free Software licenses (e.g., in the wide adoption of GNU/Linux systems)

Anything you would like to add or that you think our audience should know?

We strongly urge you to check out The Free Software Foundation website, which should clarify many of the questions you have regarding "Free Software" in general:

http://www.fsf.org/licensing/essays/free-sw.html.

And they said it couldn't be done.

Ada Inside

UK — NextGeneration Air Traffic Control Systems

From: AdaCore Press Center
Date: Tuesday June 19, 2007
Subject: GNAT Pro Chosen for UK's Next Generation ATC System
RSS: www.adacore.com/category/press-center/feed/

NEW YORK and TAMPA, Fl., June 19, 2007 — Systems & Software Technology Conference — AdaCore, provider of the highest quality Ada tools and support services, today announced that Praxis has selected AdaCore's GNAT Pro for the implementation of the UK's next-generation Interim Future Area Control Tools Support (iFACTS) air traffic control system for its client NATS.

iFACTS will use a new program that is being designed and implemented from the start with the SPARK Ada language, a choice based on Ada's proven strength in developing large, long-lived, high-reliability systems. The program will be using the GNAT Pro native toolset on IBM AIX workstations as the development environment. AdaCore's unparalleled support is one primary reason that Praxis chose AdaCore. AdaCore provides state of the art Ada compilation systems and support for this and many other native and embedded platforms.

iFACTS will provide Air Traffic Controllers with a set of advanced tools to increase capacity to meet the growing demand from the civil aviation industry. It will also alert Controllers to flights which are not following their flight plan and detect medium term conflicts, which will also enhance safety capability.

Keith Williams, Praxis' Managing Director, said, "It is extremely exciting to be able to deploy our capability in critical software on the iFACTS project and work with partners who combine advanced technology with rigorous safety certification — AdaCore was the perfect solution for our high integrity SPARK Ada development needs."

"iFACTS is the future of air traffic control," said Robert Dewar, President and CEO, AdaCore. "The combination of Praxis' experience in critical systems engineering and the high integrity of SPARK Ada enables the development of this vitally important and sophisticated system."

NATS has pioneered research and development of advanced air traffic control tools for several years from its simulator and research centre at Hurn. The iFACTS project will deliver a subset of these tools onto the system at the company's main en-route Control Centre at Swanwick in Hampshire. Currently undergoing trials, iFACTS will be installed at the London Area Control Centre, Swanwick. Following full development, training, and installation of a new workstation at Swanwick, iFACTS will be introduced into service.

About Praxis

Praxis is a systems engineering company specialising in safety- and mission-critical applications. Praxis leads the world in specific areas of advanced systems engineering specifically: ultra low defect software engineering, safety engineering for complex or novel systems, and tools/methods for systems engineering. Praxis offers clients a range of services including turn-key systems development, consultancy, training and R&D. Key market sectors are Aerospace, Defence, Air Traffic Management, Railways and Nuclear. The company operates internationally with active projects in the US, Asia and Europe. UK offices include London and Bath. It is wholly owned by Altran Technologies which is a global leader in innovation engineering and employs 16,000 engineers across the world. www.praxis-his.com

UK — SPARK Ada for Thales

Praxis wins contract to supply SPARK Ada for Thales aircraft software system

30 August 2007 — Praxis has won a contract with Thales UK, Air Operations in Wells, to supply the SPARK Ada toolset as part of ongoing product development by Thales in Aircraft Mission management and Mission Planning. Thales UK specialises in developing high integrity software for evaluation and validation of mission plans and to ensure that this software is developed to the required standard, SPARK Ada is used as part of a rigorous development process. One of the primary goals was to select a tool with a proven reputation in the Safety community. The SPARK tool met all of the requirements and is now being used on a major programme.

SPARK provides a programming language and verification environment for

high-integrity software. The core SPARK language combines an unambiguous subset of Ada with annotations or "contracts" that allow wholly static verification of key program properties such as information-flow, absence of run-time errors, program correctness, and invariant safety and security properties. The toolset offers a combination of soundness, completeness, efficiency and analytical depth which is unmatched by any other language. SPARK also meets all known regulatory requirements and standards for high-integrity software.

"We are very pleased that Thales UK has selected SPARK for this programme" said Rod Chapman, SPARK Products Manager and Principal Engineer at Praxis. "This win further re-enforces SPARK's position as the de-facto choice for software where high-integrity or ultra-low defect rates are required."

Successful Ada projects

From: Ed Falis <falis@verizon.net>
Subject: Re: Current status of Ada?
Newsgroups: comp.lang.ada
Date: Sun, 26 Aug 2007 18:46:17 GMT

[…] there appears to be a growing recognition within the Open Group that Ada has a place for safety and security sensitive systems. It's going to take a long time for it to be completely flushed away. As you said, there are a surprising number of programs still using the language.

From: Ed Falis <falis@verizon.net>
Date: Sun, 02 Sep 2007 20:05:05 GMT
Subject: Re: Current status of Ada?
Newsgroups: comp.lang.ada

And for another perspective that I consider valuable check out "Lean Software Strategies" by Middleton and Sutton. It won the 2007 Shingo award, which Business Week has called the "Nobel Prize" of manufacturing. They recommend Ada not for its technical merits, but for its role in ensuring the integrity of the software production process. An interesting point of view that may resonate with some of you here.

From: Rod Chapman <roderick.chapman@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Sun, 02 Sep 2007 21:29:25 −0000

Indeed it did. Jim Sutton (of Lockheed) was one of the designers of the software process that's used on the C130J Mission Computers. Which programming language do they use? Yup… SPARK… :-)

From: Rod Chapman <roderick.chapman@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Tue, 28 Aug 2007 07:58:35 −0000

> There is no major Ada project that is visible to the larger community of software developers.

You don't consider iFACTS to be a "major" Ada project? Perhaps you don't think it counts because it's based in the UK?

> At present, I am the last hold-out for keeping Ada in some small part of our curriculum.

I would suggest keeping SPARK on the curriculum and just quietly forget to tell your colleagues that it's Ada… :-)

I can think of one US government agency that's very interested in having faculty teach strong software engineering, static verification, formal methods and so on: the NSA. We have several such universities doing so right now, using SPARK as the primary vehicle.

From: Richard Riehle <adaworks@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Wed, 29 Aug 2007 05:23:09 GMT

OK. iFACTS is a major project. However, it is not very visible in the U.S. Agree about JSF. However, the decision to use C++ was a bit insane.

NSA might be actually using Ada, or it might be simply exploring it. If they are using it, some of my former NPS students who are now at NSA might be in the picture somewhere. However, I'll never know that since they abandon all contact once they are shackled to their cubicle at Ft. Meade [NSA headquarters —su].

I am trying to keep an active interest in SPARK. There are a few professors in our formal methods area who have an interest in SPARK and when you next visit NPS, I'll make sure you have a chance to present a little seminar for them.

From: Maciej Sobczak <maciej@msobczak.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Tue, 28 Aug 2007 04:46:38 −0700

[…] I might not be the most informed in the subject, but I have an impression that Ada is currently better supported in Europe than in US. Some French universities use Ada quite heavily. I also have some signals from other European countries where students choose Ada as a vehicle for their automatics projects.

From the "spectacular projects department", high-speed trains come to mind. Of course I mean — European high-speed trains.

The last conference on Ada (and thereabouts) was held in Geneva.

Some of the frequent posters on this group work in Europe as well and it looks like they are using Ada at work.

Projects like AWS or PolyORB seem to have European origins.

In short — the fact that US military industry turns away from Ada is at most the US problem, not the Ada problem in general. Nothing to fuss about.

Just my 0.05 Euro. ;-)

BTW: Ada Lovelace was European as well…

Indirect Information on Ada Usage

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

Job Description: United Arab Emirates

Ada Application Developer — United Arab Emirates

[…] looking for creative Ada 95 Application Developers to work in the United Arab Emirates to assist our customers as they solve complex business problems in unique technology environments.

We are currently seeking experienced Ada 95 software engineers who are personable and able to work independently on a small team.

Must be familiar with Rational Apex, Java, C and Sybase.

Salary up to $65 hourly plus living expenses depending on experience.

Skills and Experience:

- Must be US Citizen.
- Ada 95 Application Developer with 3 or more years experience required.
- Experience must include: Ada 95, Rational Apex, Java, C, Sysbase
- B.S. in Computer Engineering or Electrical Engineering from a leading engineering school such as Rose-Hulman preferred.

Job Description: India

[…]

Essential:

1. Ada programming hands on (At least he/she has done programming for 1.5 to two years)
2. UML Design hands on (At least he has done design for 1.5 to two years)
3. Avionics domain (usage of DO and ARINC standards)

Added advantage:

1. Flight Management Concept (FMS)
2. Rhapsody in Ada/ C

Positions available with a Multinational co based in Guindy.

Job Description: France

[…]

Mission:

Implementation of software engineering and numeric techniques for the

development of software for distributed and mobile systems.

Activities:

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

Skills:

- Knowledge and experience in systems and networks: Unix, TCP/UDP, SQL
- Knowledge and experience in programming: C, Java, (Ada appreciated)
- Knowledge and experience in distribution technologies: CORBA, RMI,
- Experience in software specification and modeling: UML, ADL,
- Experince in algorithms for distributed systems: global state, distributed memory
- Experience in verification: petri nets, synchronous languages
- Degree equivalente to a Master in informatics

Context:

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

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

[Translated from French. —su]

Ada usage in secret projects

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Tue, 21 Aug 2007 17:29:01 −0500

[See original post for references —su]

> I am conducting market research regarding the amount of Ada code in active use today. […] This estimate should include, and emphasize if necessary, legacy code that was written more than ten years ago and is either being used as-is or with minor modifications.

The AdaIC took a survey of Ada usage in 2005. Since the data was self-reported, its hard to know how accurate the results are. There is a presentation with a summary of the results at [1] and a article with results at [2].

From: Jeffrey R. Carter <jrcarter@acm.org>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Wed, 22 Aug 2007 00:53:12 GMT

[See original post for references —su]

> And it is fair the say that the survey under-reports as many people working DoD related projects get enough briefings about violation of export laws that they are not going to risk their jobs to answer a survey that asks for specific project names.

And some projects using Ada commercially consider it a competitive advantage and keep it secret.

Richard Riehle made a similar inquiry about new Ada projects recently (May 03) and might be able to give you some additional information. See [3].

From: Markus E Leypold <kontakt@m-e-leypold.de>
Date: Thu, 23 Aug 2007 10:13:13 +0200
Subject: Re: Current status of Ada?
Newsgroups: comp.lang.ada

> […] It seems to me it's in the companies' best interests to say whether they are using Ada, since Ada developers are tricky to get hold of. Keeping it secret seems to me both difficult (what do you write in the job ads?) and counterproductive.

Depends what you're offering. If you're offering a program component it might be in your best interest to offer it in a language the customer uses or might want to use (and if your customer works in defense or aerospace that might well be Ada). If you're offering a complete application it depends on the customer: I've been bidding for projects where the customer then went for getting the thing done in C# (by someone else). I've actually had other projects where the customer didn't care too much (because he'd have to buy maintenance from a third party anyway). I suggest that in the latter case it's often more useful to focus on the properties / features of the system to be developed, not on the implementation technique or the language: That would only serve to confuse the customer (looking up Ada in "the Internet" could even give them the impression that they are getting sold a dead end technology and now amount of "arguing" will server to clean up this impression: You won't get the opportunity to argue very much at all). I wouldn't keep the use of Ada secret in this cases, but neither would I try to dilute my sales pitch by introducing the irrelevant question of which technology will be actually used: The important bottom lines are features (including stability, freeness from software defects and this like).

It rather depends on the given situation whether commitment to a certain language or development method is a competitive advantage. […]

From: Richard Riehle <adaworks@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Sun, 26 Aug 2007 10:51:38 −0700

I once had a commercial client that required a non-disclosure agreement about their use of Ada because of competitive reasons. In their view, their competitors would use this fact against them as a sales gimmick. The fear was that the competitors would ridicule them for "using a language that was not part of the mainstream and had been rejected by the Department of Defense." […]

As noted in an earlier post, I made an inquiry some time ago about the current state of Ada usage. I am constrained from publishing the names of projects that are using Ada, but I was surprised to find that there are still quite a few. Unfortunately, such constraints do not help to promote the awareness that Ada is real and continues to be a valuable tool for building software systems. I promote it whenever I can for my own students and have had thesis students do their M.S. thesis using Ada. I make it clear in all of my software engineering classes that Ada continues to be the most effective language when one needs to take an engineering view of the software process.

But individual professors of computer science are of little importance in the effort to improve the state of Ada utilization and awareness. We need some kind of larger effort. The Ada Resource Association (or whatever it is currently called) has proven ineffectual. The AdaIC web site, while in capable hands, has no pro-active role. And the Ada compiler publishers seem to be ashamed to admit, broadcast, or let anyone know that they have Ada products. When is the last time that Rational had any information about its Ada compiler at a conference or trade-show? When is the last time that any Ada compiler publisher had a booth at a trade-show? When have we last seen any publicity about the value of Ada for some major project? Where has anyone seen an Ada textbook for sale in a bookstore? Even the computer-centric bookstores have no books on Ada — none.

As long as Ada remains invisible the number of projects will decline. As long as officials in the DoD believe that Ada is not supposed to be used for military projects anymore (many believe just that), Ada will be in decline. […]

From: Maciej Sobczak <maciej@msobczak.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Wed, 22 Aug 2007 01:44:10 −0700

> And it is fair the say that the survey under-reports as many people working DoD related projects

[…]

It is interesting, but the problem is really in what kind of information this survey is expected to provide. For the sake of mental experiment, let's assume that 50% of Ada projects are classified. At first sight, the survey under-reports by 50%. But if the survey is supposed to provide some insight on how strong and vibrant is the Ada community (for example, the requester wants to be sure that she will not be left alone with her problems), then the survey is 100% exact, because it comes from those contributions that actually form that vibrant and responsive part of the community. Anybody else is effectively out of the community.

It is more or less analogous to the report that says that we might have fuel problems within the next N years. Does it under-reports the reality considering the fact that there are whole *planets* in our solar system that are composed almost entirely of methane or hydrogen? They might be somewhere there to look at, but are effectively unreachable in the same N-years time frame, and so are completely useless in this context.

What I want to say is that every time a similar question is raised, the Ada community tries to "pump up" or visually inflate itself by mentioning some nebulous DoD or otherwise classified activity. This is cheating — the Ada community is effectively as strong as the number of people that are able and willing to talk about their experiences. Everything that is outside of this is just as useful as hydrogen on Jupiter.

From: Jeffrey Creem <jeff@thecreems.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Wed, 22 Aug 2007 12:15:01 GMT

If the purpose of the study is what you say then I totally agree. If the purpose of the study is to see how many projects/Lines of code are being done in Ada to determine if there is enough activity to support the various vendors so that 'the community' is not left in the cold by lack of vendor support, then I would assert that these surveys fail.Note the projects in question don't even need to be classified. Public release of almost any information can cause problems in big organizations.

From: Steve Marotta <smarotta@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Wed, 22 Aug 2007 15:33:34 −0000

[…] I should clarify the nature of my interest in Ada usage. I am not particularly interested in how much Ada is being used in brand-new code. I am more interested in knowing how much legacy Ada code the DoD or other government agencies are sitting on, either maintaining or using as-is.

Ada in Context

Unit Checking and Ada

From: Martin <martin.dowie@btopenworld.com>
Newsgroups: comp.lang.ada
Subject: SI Units — has Ada missed the boat?
Date: Sun, 08 Jul 2007 08:13:39 −0700

One of my pet hopes for Ada 2005 was that it would include some method of automatically checking systems of units at compilation time or with minimal run-time checking. Alas it was voted down due to time pressures and technical issues (see http://www.ada-auth.org/ai-files/minutes/min-0403.html#AI324 and http://www.ada-auth.org/cgi-bin/cvsweb.cgi/AIs/AI-00324.TXT?rev=1.3).

C++ now has the Boost library (see http://svn.boost.org/trac/boost/browser/sandbox/units) with zero-runtime cost (or at least when optimisation is switched on).

And now Java has a proposal for a similar beast (see https://jsr-275.dev.java.net/files/documents/4333/34956/jsr-275.pdf).

I've been playing around with the C++ Boost library and it seems quite good — at least for the sort of things I would use it for (embedded avionics).

Is anyone still working on an Ada solution to this?

From: Kevin Cline <kevin.cline@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: SI Units — has Ada missed the boat?
Date: Wed, 18 Jul 2007 00:11:12 −0700

A basic but usable implementation is obvious to anyone who understands C++ template metaprogramming. Of course a fully general implementation is considerably more work.

This is rather unwieldy since Java does not allow operator overloading. Who wants to read code like:

```
time_aloft =
    sqrt(height.times(2).divide(g))
```

Nor can Java check the correctness of that computation at compile time.

A satisfactory solution seems impossible until Ada allows implicit generic instantiation, and then it would be relatively simple.

Date: Mon, 09 Jul 2007 10:55:53 +0200
From: "Grein, Christoph (Fa. ESG)" <Christoph.Grein@eurocopter.com>
Subject: WG: SI Units — has Ada missed the boat?
Newsgroups: comp.lang.ada
Organization: Eurocopter Deutschland GmbH

> One of my pet hopes for Ada2005 was that it would include some method of automatically checking systems of units at compilation time or with minimal run-time checking. Alas it was voted down due to time

I have to admit that I was the main perpetrator who killed that proposal.

> Is anyone still working on an Ada solution to this?

I don't think so, if you mean a method to include in the Ada standard.

But there are a lot of such methods distributed in some home pages, see Dmitry Kazakov, mine, … (I guess there are plenty others)

I don't know C++ and so I do not know how it deals mit rad, Newton Meters and Joule in

Work [Joule] = Torque [Newton*Meter] * Angle [rad]

Or sin x = x + x**3/3! + … where x is in rad, sin x has dimension 1 (not rad).

Or Bq = 1/s, Hz = 1/s, but Bq ≠ Hz.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: WG: SI Units — has Ada missed the boat?
Newsgroups: comp.lang.ada
Date: Mon, 9 Jul 2007 11:49:15 +0200

> I have to admit that I was the main perpetrator who killed that proposal.

Fortunately. Be praised for that!

> I don't think so, if you mean a method to include in the Ada standard.

Yes, and if we'd put together more or less obvious requirements of how an Ada™ solution should look like, we would easily see a need in some language changes. My strong impression is that these changes should go beyond sole units support. For example, it is quite obvious that the unit constraint should be put on the object, be it a floating-point number or a matrix of. We just don't have any universally working mechanism for that. Same with dimensioned literals, which would imply a user-defined literals support, etc.

In short, units is just a use case, it should not be viewed as a language feature.

The new version of my measurement units for Ada with GTK tree view cell renderers and combo boxes for dimensioned values is coming soon.

> I don't know C++ and so I do not know how it deals with radians, Newton, Meters and Joule in …

Such differences should be handled outside the unit system. It cannot know the semantics of the values. So rad = 1 SI. Or else m (of height) ≠ m (of distance on a highway)

But how C++ handles (I guess it does not)

```
X := Vector (I);  -- Both X and Vector are dimensioned
Y : Unit := Ask_User_For;
Z : Magnitude := 5 dB; -- May I have logarithmic scales?
```

From: Hyman Rosen <hyrosen@mail.com>
Newsgroups: comp.lang.ada
Subject: Re: WG: SI Units — has Ada missed the boat?
Date: Tue, 10 Jul 2007 01:26:08 GMT

Here's a description of implementing units using the Boost Metaprogramming Library.

<http://www.boost.org/libs/mpl/doc/tutorial/dimensional-analysis.html>

I expect that most C++ implementations will follow the rule that 90% of the way is enough, and not bother with radians or trying to distinguish between becquerel and hertz or Fahrenheit, Celsius, and Kelvin. But if someone wants to go the whole way, they can do it. It's just a matter of writing more complicated combining rules for the templates. And none of it has any runtime overhead at all — objects of dimensioned types take no more space than plain numbers.

Date: Wed, 11 Jul 2007 21:27:30 −0400
From: Joe Simon
Newsgroups: comp.lang.ada
Subject: Re: WG: SI Units — has Ada missed the boat?

Back in the 1980s I was working on a simulator, where we the software people wrote the infrastructure and the various subject area specialists wrote the code for the things they wanted to simulate.

We (software weenies) created a package (wish I could remember the name of it) that provided basic types LENGTH_UNITS, TIME_UNITS, SPEED_UNITS, DISTANCE_UNITS, TEMPERATURE_UNITS, ANGLE_UNTIS, etc. and then defined all of the overloaded operators to convert between those.

The upshot of this was the converse stuff got so large and complex that we wrote a program that would read a text file that would define the conversions i.e. LENGTH_UNITS = SPEED_UNITS * TIME_UNITS… etc, the program then was able to define the base conversion and the related conversions (SPEED_UNITS = LENGTH_UNITS / TIME_UNITS), etc. I don't remember the details but there was also a way to define the conversion from various units into the generic _UNITS. For instance for LENGTH_UNITS, METERS would = 1, feet would be whatever the conversion from feet to meters is, furlongs would be defined similarly.

This program would then write the Ada specs and bodies for the conversion.

Down side:

This was in 1986ish. The "UNITS" package took about 4 hours to compile (on a VAX using VAX Ada) and every component took a long time to compile because every expression had to be compared to the myriad of overloads to determine if the expression was valid.

If a conversion didn't exist, you added the appropriate info to the conversion file (and hope you did it right), checked it back into CM, and the build ran overnight. The next morning you could compile your expression.

Up side:

We NEVER had unit conversion issues, as the unit analysis was done by the compiler.

This enabled the subject matter experts to write code that defined a value of type say mytime : TIME_UNITS, and myspeed : SPEED_UNITS, mydistance : LENGTH_UNITS (sorry rusty Ada syntax) and then mytime := FROM_FOTNIGHTS (10.0) ; myspeed := FROM_METERS_PER_SECOND (100.0); mydistance = myspeed * mytime;

PRINTLN (TO_FURLONGS (mydistance)) ; and myint : INTEGER ; myint = 10 ; mydistance = myspeed * myint ; would not compile.

You can see why the conversion package got so big… it had to define ALL valid conversions.

Sorry this was so long. Wish I could remember more of the details. It was one of the most fun Ada jobs I worked on.

From: Dr. Adrian Wrigley <amtw@linuxchip.demon.co.uk>
Subject: Re: WG: SI Units — has Ada missed the boat?
Newsgroups: comp.lang.ada
Date: Fri, 13 Jul 2007 22:20:32 GMT

Interesting.

I tried out something similar a while back, and the method was to add the overloaded functions only when they were needed. I found that there were rather few overloaded functions needed in practise. This is because expressions tend to be rather stereotypical — exotic combinations of dimensional expressions never arise. I think my solution used generic instantiations, in the end, cutting the amount of source.

An interesting project which does include full units and dimensions support is Sun's Fortress:

http://research.sun.com/projects/plrg/Fortress/overview.html

This is very ambitious and complete in scope — at least for parallel, concurrent scientific codes.

What do people here think of their efforts? I don't see Ada mentioned in their white papers, even although there's a lot of overlap in requirements.

One interesting feature of Fortress is that "for" loops execute in any order (or none), by default. This gives the very fine grain parallelism opportunities I was talking about here a while a go.

I think any general purpose language should come with proper physical types — I'm surprised that it's normally left out. (VHDL was the first language I used which had them)

As regards Ada and units/dimensions, I think the way forward for most users is to put the units/physical types into a package and add overloadings by hand when needed. More intensive use warrants the use of generic packages. It's easy and the incremental effort is low.

If you want a comprehensive approach, write an Ada pre-processor to extend the language with a new syntax for dimensional types. Use something like ASIS as a front end, and spit out standard Ada. One way would be to check dimensional consistency and then generate the necessary package(s) with the particular types and functions needed, adding "with" clauses where necessary.

Number crunching in Ada

From: Nasser Abbasi <nma@12000.org>
Newsgroups: comp.lang.ada
Subject: Re: Interested about number crunching in Ada
Date: Wed, 15 Aug 2007 23:43:02 −0700

[See original post for URLs —su]

> I have stumbled upon Ada 95 and I have found that a recent addition was made to the language standard [1]. An addition I, a student of scientific computing, are highly interested in.

> What is the best online resource to get into the core of the new high performance vector and matrix features? Does there exist some book (yet) which covers this area? Or any other field which might be related to me (concurrency, Fortran bindings etc.)? I know C and Pascal good and I have a good start into Fortran 90/95.

> I applicate your time and help. I hope that, with a push in the right direction I will be a productive "Ada numerics hacker" in a near future.

I am also interested in this subject. Check Numeric Annex for Ada 2005 [2]

It seems to have support for Vector and Matrix objects, and the following operations: (there is a version for real and complex)

function Unit_Vector
function Transpose
function Solve
function Inverse
function Determinant
function Eigenvalues
procedure Eigensystem
function Unit_Matrix

This is an old paper called "Can Ada replace FORTRAN for numerical computation?" published in 1981 ! [3]

Dr. Martin J. Stift, uses Ada for Astrophysics [4]

Here is some Finite elements code in Ada [5]

Just few days ago, I also wrote short Ada program (even though my Ada is VERY dusty as I use Mathematica mostly these days and also Matlab and Maple), I wrote an Ada program to solve a simple second order ode using finite elements using the new Ada2005 Solve function. I wrote the same code in Mathematica and then in Ada (and also in Maple). Item #6 on this page below. It worked great and was very fast as expected. One nice thing about Ada as always, is that once one gets a clean compile, most likely than not, the code will run without problems. With other languages/systems, this is not the case. With the Ada program, once I get a clean compile, that was it. Using the other systems, I had to spend more time debugging run time errors and go back fix the code, and run again and fix errors, etc… So the Ada program was completed much faster than the others at the end. [6]

If you google around, you'll find some Ada package for matrix/vector operations and more scientific code in Ada (such as fast Fourier transforms, etc..)

I think Ada as a language is great for numerical and scientific programming. These were number of discussion on this vs Fortran on the net, check this one thread: [7]

I just do not think the current Ada 2005 numeric annex contain enough functionality.

One can always link to BLAS and linpack/lapack libraries (which are written in Fortran), I just googled around for Ada binding to blas, here is the link [8]

Here is a question I have: Why is there no standard binding to all of these libraries (blas, linpack, lapack) as part of the standard? or is there? Will Ada numeric annex be extended to do that? I think the current Numeric annex is too small.

I have no idea why any one would choose C or C++ over Ada for numerical work. It is simply beyond my understanding.

I can understand one choosing Fortran over Ada, simply due to the inertia that Fortran has in this domain, and the huge amount of existing Fortran code out there. But from a language point of view, I think Ada is definitely better for numerical work than Fortran, but having a better language is not enough in the real world.

From: Jerry <lanceboyle@qwest.net>
Newsgroups: comp.lang.ada
Subject: Re: Interested about number crunching in Ada
Date: Fri, 17 Aug 2007 02:43:01 −0700

The above binding [8] is written for Ada 95. It seems to me that a binding for Ada 2005 would have to be different from this one in that it would use the official types for vectors and matrices, that is,

```
type Real_Vector is array
      (Integer range <>) of Real'Base;
type Real_Matrix is array
      (Integer range <>,
       Integer range <>) of Real'Base;
```

as defined in Annex G.3, whereas the Ada 95 bindings at the link above use these definitions:

```
type Vector is array
      (Positive range <>) of Float;
pragma Convention (Fortran, Vector);
type Matrix is array
      (Positive range <>,
       Positive range <>) of Float;
pragma Convention (Fortran, Matrix);
```

which are declared in the user's program.

(Similarly for complex vectors and matrices.)

From: Jerry <lanceboyle@qwest.net>
Newsgroups: comp.lang.ada
Subject: Re: Interested about number crunching in Ada
Date: Thu, 16 Aug 2007 15:55:06 −0700

The new numerical aspects of Ada (Annex G.3) are excellent, providing a number of types and function overloads. The new facilities are rather basic as far as actual algorithms, but see a very recent discussion regarding linking to BLAS and LAPACK, if your installation doesn't already do that. (It seems that BLAS and LAPACK are quasi- officially recommended — the Ada designers weren't foolish enough to ignore these venerable numerical packages.)

More broadly as to the appropriateness of using Ada for numerical work, I personally haven't run across a better solution. I'm a relatively new user of Ada and am stunned at how well it works for numerical work. I have used Fortran, Pascal, Matlab/Octave, Mathematica, Maple, Igor Pro, and some others too obscure to mention or remember. Ada tops them all for programming. (Mathematica, Maple, Igor Pro e.g. have many other reasons to recommend them.)

What I (and many others) have done is to write some overloaded procs and functions to handle vector-matrix things and whatever other structures your work requires (For example, vectors and matrices of transfer functions for signal processing and control systems.) With a few overloaded functions, you can write concise yet clear code that Matlab aspires to but doesn't entirely succeed at. And you can do better than Matlab thanks to Ada's strong typing. If you have a vector x, Matlab will not allow you to compute 1.0/x but Ada will (with an overload).

I'd be glad to share my collection of overloads that allow mixing arithmetic between Integers, Long_Floats, Complex, and real and complex vectors and matrices. I know that there are a lot of combinations to fully flesh out all of these, but I've found that not all are required; and if I run across one that I don't have yet, it's just a couple of minutes to write it.

From: Jerry <lanceboyle@qwest.net>
Newsgroups: comp.lang.ada
Subject: Re: Interested about number crunching in Ada
Date: Fri, 17 Aug 2007 02:52:07 −0700

[See original post for URLs —su]

> Speaking on the subject, Numerical Recipes, considered by some as the *reference* for this subject, has releases the 3rd edition of this famous work. A huge and large book.

> And it is that is written in nothing else but C++ !

Fromhttp://www.nr.com/aboutNR3book.htmlit says: "Its code is wholeheartedly object oriented, demonstrating diverse techniques for using the full power of C++."

> I bet the next version will be Numerical Recipes in Java.

> Then after that a Numerical Recipes in C#?

> I am waiting for Numerical Recipes in Visual Basic to come out, and may be also a version in JavaScript, and why not Numerical Recipes in Perl? :)

> I guess the authors found that Fortran is no longer 'popular' enough, and C++ is the more sexy language now for selling more copies of the book.

> I think a version in Ada will be great, but of course we know that Ada is not sexy or popular enough, so I am sure this will never happen.

The old Fortran version of the book of course had a Pascal appendix and an associated smaller book dedicated to Pascal. (I have both.) Recent digging around on the official Numerical Recipes web site reveals that the later versions of the software were never made available in Pascal. (I could be partially wrong--was the original version of Numerical Recipes converted to another flavor of Pascal other than basically the Jensen and Wirth flavor?)

Anyway, the Pascal-to-Ada converter p2ada is said to have successfully converted the entire Numerical Recipes into Ada. Look here: [9]

Exceptions and out parameters

From: Adam Beneschan <adam@irvine.com>
Newsgroups: comp.lang.ada
Subject: Re: Exceptions and out procedure arguments (using GNAT GPL)
Date: Mon, 18 Jun 2007 08:44:57 −0700

> Consider a procedure that starts like this:

> procedure My_Procedure (O: out integer) is

> begin

> O := 999;

> raise My_Exception;

> […] and has an exception handler

> exception

> when My_Exception =>

> null;

> Is the routine which calls My_Procedure guaranteed to get a value […]?

No. When My_Procedure completes abnormally, due to an exception raise, X should have the value that it had before My_Procedure was called. An Ada compiler that causes X to be 999 here (if it wasn't 999 before) is incorrect. This isn't a matter of "is the compiler allowed to optimize" or "all bets are off" or "you can't rely on the value"; rather, the semantics *require* that X be unchanged. This is because O is a by-copy parameter (6.2(3)), which means that inside the subprogram, O denotes a *separate* object from X (6.2(2)), and O is copied back to X only on *normal* completion of the subprogram (6.4.1(17)), but an exception raise causes My_Procedure to be completed abnormally (7.6.1(2)).

By-reference parameters work differently. If, for example, your OUT parameter were a tagged record, and you had assigned a component of it to 999, it should still be 999 even after the exception in My_Procedure is raised. As I interpret the rules in 11.6, this might not be the case if the exception raise is due to a language-defined check; if, after you assign the component to 999, you do an array access on a nonexistent element, so that Constraint_Error is raised, this is a language-defined check, and now I think the compiler may be allowed to optimize in a way so that the assignment of the component to 999 might not take place. But in your example, you have an explicit raise of a user-defined exception, and 11.6 doesn't apply to those, as I read it. The same would apply in an access parameter case; if My_Procedure is abandoned due to a raise of a user-defined exception, you can count on any assignments that you've already done through the access value, but you can't count on assignments done before My_Procedure is abandoned due to a language-defined check.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Exceptions and out procedure arguments (using GNAT GPL)
Newsgroups: comp.lang.ada
Date: Tue, 19 Jun 2007 22:07:59 +0200

> My gut feeling is that, in the abstract, a subprogram should either produce a result *or* (perhaps) raise an exception, but not both; in general, if your definition of a subprogram is that, under certain conditions, the subprogram will raise an exception AND the caller can expect a certain value to be returned (whether in an OUT parameter or an IN OUT or a global or in something pointed to by an access parameter or whatever) even though an exception is raised, the design is wrong. It's better to use an OUT status code of some sort in that case.

I don't think this is a good advice. In my view a right design assumes that whether an exception is propagated or not, the subprogram should not leave anything in an undefined state.

That is independent on the way an in-out parameter is passed. If the parameter is by-reference, then the subprogram shall document all side effects on it, which cannot be rolled back.

(Thee is a quite specific case of non-initialized out parameters, but I don't like the idea of using that pattern rather than result anyway.)

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: Exceptions and out procedure arguments (using GNAT GPL)
Date: Tue, 19 Jun 2007 16:40:31 −0500

> I eliminated the problem by modifying the offending procedure to not raise exceptions. It now returns a status code in an additional out argument.

My initial reaction to this was that it is like cutting off your foot because your toe itches. ;-)

But I do have to agree with Adam that there is something wrong with the design if you are expecting to get results back even if an exception is raised. That does seem to be inappropriate use of an exception; it's not an error at all if you expect results (it's just another normal case).

Still, in general result codes make me ill, so I would be at least as concerned about a design that seems to be combining two operations (one to return the initial results, and one to make the checks that lead to errors).

Representation issues

From: petter_fryklund@hotmail.com
Newsgroups: comp.lang.ada
Subject: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
Date: Tue, 19 Jun 2007 06:16:53 −0700

The Size of the following record is 112 on Linux, but 104 on Solaris.

```
type Something is record
   A : Packed_16;
   B : Packed_8;
   C : Packed_8;
   D : Packed_8;
   E : Packed_32;
   F : Packed_32;
end record;
pragma Pack (Something);
```

Where

```
type Packed_Byte is mod 2 **8;
for Packed_Byte'Size use 8;
type Packed_Bytes is
        array (Natural range <>) of
        Packed_Byte;
for Packed_Bytes'Alignment use 1;
for Packed_Bytes'Component_Size use
        Packed_Byte'Size;
type Packed_8 is new
        Packed_Bytes (0 .. 0);
for Packed_8'Size use 8;
```

and similar. Why is pragma Pack ignored?

Newsgroups: comp.lang.ada
Subject: Re: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
From: Georg Bauhaus <rm.tsoh+bauhaus@maps.futureapps.de>
Date: Tue, 19 Jun 2007 16:02:13 +0200

I don't think pragma Pack is ignored, there are "should"s in the RM and then alignment. You might need a rep spec, though.

From: petter_fryklund@hotmail.com
Newsgroups: comp.lang.ada
Subject: Re: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
Date: Tue, 19 Jun 2007 07:15:41 −0700

We used to have rep spec's very early in a previous project, but one of the other team members found that pragma Pack did the work as well for Solaris, so we decided to do without rep specs. We thought it was a good decision when the number of similar records increased over 500. Now we are not so sure anymore. The before mentioned team member thinks it is a bug and will probably submit a bug report to ACT.

As a former UNISYS employee, word size is always 36 ;-)

From: Bob Spooner <rls19@psu.edu>
Newsgroups: comp.lang.ada
Subject: Re: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
Date: Wed, 20 Jun 2007 11:51:27 −0400
Organization: Penn State University, Center for Academic Computing

Are the processors different on the two systems? For some processors, the overhead for retrieving a 32-bit value that is not aligned on an even address or an address divisible by four could be enough higher that the default record layout will be different from what you experienced with Solaris. Then, depending on the interpretation of "storage minimization should be the main criterion when selecting the representation of a composite type" for pragma Pack in the RM, the packed layout may be different as well, since pragmas are only advice to the compiler. If you use a record representation clause, I would expect that if the compiler could not generate code to give you what you asked for, it would generate an error. You can use the 'Size attribute to specify the overall record size, but then the compiler is still free to rearrange the order of components to optimize storage and retrieval of the record components. For exchanging binary data between heterogeneous systems, my experience has been that record representation clauses are necessary to insure that the data representations are identical.

From: Adam Beneschan <adam@irvine.com>
Newsgroups: comp.lang.ada
Subject: Re: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
Date: Wed, 20 Jun 2007 09:35:58 −0700

> For exchanging binary data between heterogeneous systems, my experience has been that record representation clauses are necessary to insure that the data representations are identical.

Absolutely. You can't count on Pack to do things a certain way; the RM says that the compiler is free to rearrange the components as it sees fit to make things smaller. Pack is appropriate when you want the type to be as small as possible but don't really care how it's laid out. It's not appropriate if you care how your record is laid out; you'll need a record rep clause for that.

From: Simon Wright <simon.j.wright@mac.com>
Newsgroups: comp.lang.ada
Subject: Re: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
Date: Wed, 20 Jun 2007 21:50:12 +0100

Or (if on GNAT) you could use a version of the compiler's stream support that uses XDR representations 'on the wire' — we transparently communicate between PowerPC and Intel hardware like this as a matter of course without having to think about it or expend any effort.

There's always a price to pay, of course; the packing isn't dense, and it can be quite a challenge to work out what bytes are actually being sent (eg, if you find yourself having to talk to C after all).

From: Bob Spooner <rls19@psu.edu>
Newsgroups: comp.lang.ada
Subject: Re: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
Date: Thu, 21 Jun 2007 10:50:55 −0400
Organization: Penn State University, Center for Academic Computing

Yes, XDR takes care of things like endianness, etc. that otherwise get in the way, but as you point out, there's always a price to be paid for generality. In some cases it looks like it even will take care of differing floating point representations, although I wonder about out of range problems when converting. Isn't there an XDR library for C? I know that there is one for Fortran. I would think that as long as you have an XDR library for the language with which you need to communicate, you wouldn't have to decode the bytes yourself; or have I misunderstood what you are saying?

From: Simon Wright <simon.j.wright@mac.com>
Newsgroups: comp.lang.ada
Subject: Re: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
Date: Fri, 22 Jun 2007 06:20:55 +0100

For us that was a big 'if'. There was a scripting language that didn't understand anything beyond bytes on the wire, so you had to be pretty conversant with the actual layouts used by the Ada. We used an ASIS-based tool to work this out, but it's not well integrated into the build process. Fortunately message definitions on the interfaces where it matters don't change very often.

Another problem was that the message header had to contain the length of the message body — the easiest way to do that is of course to construct the body first! (or perhaps a dummy message body during program initialization/elaboration?)

From: Ole-Hjalmar Kristensen <ole-hjalmar.kristensen@sun.com>
Subject: Re: pragma Pack does not work on GNAT 5.01a for Redhat Linux.
Date: 07 Aug 2007 12:51:25 +0200
Newsgroups: comp.lang.ada

Sorry for any confusion. My point was simply that XDR has been a de facto standard available to C programmers for a very long time. If you have an XDR library for Ada, you can exchange messages between C and Ada programs with no problems, since the protocol is described in terms of XDR, and not tided to any specific language.

Portability and System calls

From: Maciej Sobczak <see.my.homepage@gmail.com>
Newsgroups: comp.lang.ada
Subject: System calls — GNAT library vs. direct bindings
Date: 31 May 2007 07:58:40 −0700

Let's suppose that the problem is a network-aware application on the POSIX system. Ada doesn't know about network per RM, so we have to look around for libraries. One option is to use the GNAT library that has some package for sockets. Another is to write thin wrappers that bind directly to relevant system calls (socket, connect, read, write, close, bind, listen, accept, select — that's it for the basic stuff).

1. The advantage of using GNAT library is portability across various systems that are targeted by GNAT. (I assume that the GNAT library is available outside of POSIX — if not, please correct me.)

2. The advantage of writing custom thin wrappers is portability across various compilers that target POSIX platforms.

The first is not really important for me, I will probably use only POSIX systems. The second is not important neither, I will probably use only GNAT.

In other words — the focus for evaluation should go elsewhere. Performance? Maintainability? Some other factors?

From: Simon Wright <simon.j.wright@mac.com>
Subject: Re: System calls — GNAT library vs. direct bindings
Date: Thu, 31 May 2007 21:07:50 +0100
Newsgroups: comp.lang.ada

Our compiler is GNAT, and we develop and unit test on Windows, the target is VxWorks on PowerPC.

We decided that GNAT.Sockets was perfectly OK for our needs (that doesn't mean we think it's perfect) and that we had enough to do developing our application without doing a socket library as well.

AdaSockets wasn't really an option given that GNAT already has a library. It would have been yet another library to justify to the technical authority.

From: Simon Wright <simon.j.wright@mac.com>
Newsgroups: comp.lang.ada
Subject: Re: System calls — GNAT library vs. direct bindings
Date: Fri, 01 Jun 2007 20:29:21 +0100

> Why is it not perfect?

Because it was created by humans …

Specifically, the behaviour with UDP streams is bizarre — it all looks as though the data is transferred correctly, but each atomic data item is sent/received in a separate datagram.

The implementation of Socket_Type (and there's a naming issue straight away!) suffers from not being controlled, and might be better reference-counted (we've just had to deal with a file descriptor leak caused by this).

I'm not sure that it was necessary to use 2 (3, temporarily) invisible sockets in the

implementation of Selectors (perhaps it is, haven't thought about it deeply).

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 31 May 2007 23:10:30 +0200
Newsgroups: comp.lang.ada
Subject: Re: System calls — GNAT library vs. direct bindings

Are standards and portability still an issue in Ada projects?

Aonix seems to offer a standards conforming POSIX binding. Can't say how much networking it offers.

From: Pascal Obry <pascal@obry.net>
Date: Fri, 01 Jun 2007 22:13:49 +0200
Subject: Re: System calls — GNAT library vs. direct bindings
Newsgroups: comp.lang.ada

> Performance — use the OS's core language. Which normally means using C instead of writing code in Ada or using Wrappers. But as the performance of the code goes up the maintainability starts to drop. This is due to the fact that performance algorithms are normally tied to the hardware and as hardware is update the algorithms may need to be rewritten just to maintain current performance levels which increases the maintainability cost.

That's nonsense! An intern project comparing the "same" application built with OpenMP/C++ and Ada shows that we have the same level of performances. Ada is slightly better in fact. This is not one-to-one kind of comparison as we wanted to use both technology as it should. So in OpenMP/C++ we have parallel loops on the Ada side we have designed a pipeline of tasks. The Ada architecture is then higher level, but gives slightly better performances.

I'm fighting days after days the idea than low-level gives better performances. This is just plain wrong. It has probably be true at some point with tailored assembly applications. This is just not possible those days as processors are so complex than only a compiler can schedule the instruction properly. So no, C based languages, as low-level as they seems, are not necessary faster.

Today discussion was about the C ternary operator being faster than a standard if. The gain is three CPU cycles (well that's what the guy was arguing). Who cares ? Especially when the algorithm is just a mess of non optimal code :(How often people get swamped by low-level stuff amaze me!

Now please this is ONE bench. It just shows that at least low level does not necessary means faster, that's all. I have not said that Ada is always faster right :)

From: Jeffrey R. Carter <jrcarter@acm.org>
Newsgroups: comp.lang.ada
Subject: Re: System calls — GNAT library vs. direct bindings
Date: Fri, 01 Jun 2007 17:42:36 GMT

It's usually better to reuse something that has been widely used than to recreate the wheel. Reuse is generally less effort than creating a new library. An existing library is usually better tested and has fewer errors than a custom library. And an existing library's maintenance is generally not part of your project's maintenance effort, while a custom library's maintenance is.

Portability is generally a good thing, even if you don't think you need it. Many systems that were created under the assumption that portability wasn't an issue were later ported at greater effort than would have been involved in creating them to be portable in the 1st place.

To my mind, given the existence of a library that is portable across compilers and platforms, such a library should be used. In the absence of such a library, but given the existence of an existing library such as GNAT's, I'd probably choose the existing library, but would add a wrapper around it in case I need to use another compiler. Only in the absence of any existing libraries would I write my own, and I'd try to make it portable across compilers and platforms.

Dynamically reallocated buffer

From: Maciej Sobczak <see.my.homepage@gmail.com>
Subject: Dynamically reallocated buffer
Date: Thu, 31 May 2007 23:33:46 −0700
Newsgroups: comp.lang.ada

I need a dynamically reallocated buffer of bytes, which I can extend at run-time by appending new fragments to the whole buffer. The purpose of the buffer is to pass it later to the subprogram that writes it "en bloc" to some external device.

For those of you who know C++ I need something like:

```
vector<unsigned char> buffer;
// fill the buffer with push_back or
      insert at end
// ...
write_to_device(&buffer[0],
buffer.size());
```

The problem is that Ada.Containers.Vectors does not provide the necessary guarantees to be any useful in this context.

What are your suggestions?

From: Matthew Heaney <mheaney@on2.com>
Newsgroups: comp.lang.ada
Subject: Re: Dynamically reallocated buffer
Date: Fri, 01 Jun 2007 07:49:09 −0700

Right, that container does not guarantee that the underlying structure is a contiguous array (as is the case in C++). Think of it not as an unbounded array (which would a physical view), but rather as a container that provides random access (which is the logical view).

In your case you'll probably have to use an unbounded array directly. If you want a vector-like thing then you can just grab the vector source code and modify it as you see fit.

Newsgroups: comp.lang.ada
Subject: Re: Dynamically reallocated buffer
From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sat, 02 Jun 2007 08:38:02 −0400

>> The problem is that Ada.Containers.Vectors does not provide the necessary guarantees to be any useful in this context.

> Right, that container does not guarantee that the underlying structure is a contiguous array (as is the case in C++). Think of it not as an unbounded array (which would a physical view), but rather as a container that provides random access (which is the logical view).

SAL.Poly.Unbounded_Arrays.
http://stephe-leake.org/ada/sal.html

Newsgroups: comp.lang.ada
Subject: Re: Dynamically reallocated buffer
From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Mon, 04 Jun 2007 20:25:59 −0400

> How would I pass the array storage of an Array_Type object to a byte copying function? (As in write_to_device(&buffer[0], buffer.size());)

That operation is not provided. You need to derive from Array_Type and provide an access function that returns the address of the current actual storage.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Dynamically reallocated buffer
Newsgroups: comp.lang.ada
Date: Fri, 1 Jun 2007 09:30:04 +0200

```
type Device_Buffer is
      array (Positiver range <>) of
Interfaces.C.unsigned_char;
```

The maximal block size is usually known. If not, then device is stream-oriented and blocks can be safely split into chucks of known size. In that case Device_Buffer is a segmented buffer. In most cases you don't need to realloc anything.

From: Maciej Sobczak <maciej@msobczak.com>
Newsgroups: comp.lang.ada
Subject: Re: Dynamically reallocated buffer
Date: Fri, 01 Jun 2007 06:08:31 −0700

I might want to pack it.

[…] in the worst case I need a list (or some other container) of Device_Buffers. Yes, that makes sense.

Date: Fri, 01 Jun 2007 13:32:19 +0200

From: Georg Bauhaus <bauhaus@futureapps.de>
Newsgroups: comp.lang.ada
Subject: Re: Dynamically reallocated buffer

I'd use references to normal arrays and if I wanted to be sure that a program can rely on realloc() I'd consider a simple storage pool using this OS function. (GNAT has System.Memory with ready made bindings on some platforms.)

Then just call new Flex_Buffer(as needed). Possibly behind some Container like interface (or Controlled and Reserve_Capacity in Initialize).

Returned constant objects

From: Alex R. Mosteo
Newsgroups: comp.lang.ada
Subject: Amount of copying on returned constant objects
Date: Fri, 15 Jun 2007 19:19:00 +0300

[…] I know that some "in" arguments may be passed as copies or as references, at compiler discretion (this is one of these things that "programmers shouldn't care about", many times quoted).

I wonder however about results of functions, that are not modified. Look for example at the Element function of the new Ada.Containers. They return the stored item, that may well be a quite large controlled tagged type, for example.

Now, many times I want to query an element just for read-only purposes. I'm faced with two options:

1) Just call Element on the container Key/Index, and be done with it.
2) Do a Find+Query_Element, which requires defining an extra procedure and somewhat breaks the flow of control, but ensures no copying.

I tend to go with 1) because of laziness and the "no premature optimization" rule. In C++ I could use constant references. Now, I wonder if

a) is there something in the ARM that prevents an equivalent transparent optimization in the Ada side (returning the reference when it is detected that the returned object is not modified)?
b) If not, do you know of compilers that do this in practice? (Specially interesting for me would be GNAT at -O2/-O3).

Failing these, I guess I could define constant accesses for use in my own functions, but I find this not very Ada-like. Any other ideas?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Newsgroups: comp.lang.ada
Subject: Re: Amount of copying on returned constant objects
Date: Fri, 15 Jun 2007 18:32:25 +0200

The "compiler discretion" does not apply to all cases. The compiler is required to pass parameters of limited and tagged types by reference. I'm not entirely sure that that also applies to the result of a function call, but that would seem reasonable. If the function returns a class-wide or other unconstrained type, then the object will have to be on the heap (in the region GNAT calls the "secondary stack").

> 1) Just call Element on the container Key/Index, and be done with it.

That would be my approach.

From: Alex R. Mosteo <alejandro@mosteo.com>
Newsgroups: comp.lang.ada
Subject: Re: Amount of copying on returned constant objects
Date: Mon, 18 Jun 2007 20:23:49 +0300

> I might want a constant copy because the object in the container is going to be modified… Can a compiler detect this?

I don't think it can, at least easily. I guess that if you're keeping a copy, constant or not, the optimization opportunity is lost. But what about short-lived objects, like…

```
if Container.Element ("key").Is_Nice
  then -- Container for some tagged type
  …
end if;
```

This is the kind of copies that I see interesting to optimize away. I don't know enough about compilers to say if it is reasonable to expect one to detect this situation or not.

If not, one possibility would be to have

```
function Element (Key : Key_Type)
      return Element_Type;
```

and

```
type Constant_Access is access
      constant Element_Type;
function Element (Key : Key_Type)
      return Constant_Access;
```

but I'm not sure about the amount of ambiguities one would get in that case. In any case this does not exist in the standard 05 containers.

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: Amount of copying on returned constant objects
Date: Mon, 18 Jun 2007 15:25:33 −0500

> Or in a more Ada 2005 way:

> function Element

> (Key : Key_Type) return access constant Element_Type;

The problem with this is that this access can be saved, and any operation on the original container could make it become dangling (and thus any further use be erroneous). That is *very* unsafe and virtually impossible to detect.

There were a substantial number of people (a group that includes me) that want the containers to be safer than using raw access types (because they can do checks that would be too tedious to do in hand-written code). That's why the containers access-in-place routines use access-to-subprograms, because they can have tampering checks that prevent the dangling access problem (you get Program_Error if you try to do something that could make the element inaccessible). That makes them much safer than returning a raw pointer.

We actually spent quite a bit of effort on trying to find a way to secure access values returned this way. But it isn't quite possible: even if you make them uncopyable; they still can be held onto long enough to potentially cause trouble with a renames.

What really would help would be a way for the container to know when the access was destroyed, but there isn't any obvious way to do that in Ada.

Dmitry might (will?) tell us that a user-defined ".all" operation would do the trick, but it's not obvious how to define that operation so that the ".all" definition itself would not expose the original problem.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Amount of copying on returned constant objects
Newsgroups: comp.lang.ada
Date: Tue, 19 Jun 2007 10:26:38 +0200

In Ada, which has no procedural types closure itself is a problem because an access type is still there (now to the procedure). Further, this approach does not work if we needed to access several elements of the same or different containers. How to do this:

```
  (Get (A, First (A)) + Get (A, Last (A))) /
2;
```

with closures in a more or less readable form?

There is a problem of complexity introduced by each new type here. There is a type of the container, there are types of the element and the index. That's already triply dispatching in the most general case. [Actually, it is far more if ranges and other subsets of index are introduced] I don't want yet another type of the access to element, or a type of the procedure to access to the element etc. It is a mess when the container gets derived from. We have no any language mechanism to bind all these geometrically exploding combinations of types together.

BTW, which problem we are talking about. There are at least two:

1. An "easy" one is Alex's example:

```
if Container.Element (Key).Is_Nice then
```

What he actually needs here is to force the compiler to infer from Element's Is_Nice a new container operation:

```
if Is_Nice (Container, Key) then
```

composed out of container's Element and element's Is_Nice.

2. A difficult one:

```
Declare
  X : Item renames Get (Container,
          Key_1);
  Y : Item renames Get (Container,
          Key_2);
begin
  Remove (Container, From => Key_3,
          To => Key_4);
  X := Y;
end; -- This must be safe and efficient
```

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: Amount of copying on returned constant objects
Date: Tue, 19 Jun 2007 16:33:26 −0500

> I guess then that some reference counting companion type (or maybe making Cursors tagged and more heavyweight) was discarded because the distributed overhead?

Cursors can be tagged if the implementation so chooses, but that doesn't have an effect on the element access problem. For that you need something that allows direct dereferencing, and in Ada as it stands, that can only be an access type.

Having a companion type would work if it was impossible to separate the access from the reference counter. But there is no way in Ada to have a visible access type that cannot be assigned out of its surrounding wrapper. (Which is why I said that Dmitry would say that redefinition of ".all" for a private type could solve the problem.)

From: Markus E Leypold <kontakt@m-e-leypold.de>
Subject: Re: Amount of copying on returned constant objects
Date: Wed, 20 Jun 2007 03:31:19 +0200
Newsgroups: comp.lang.ada

I wonder whether you can't hide the dereferencing. I had a similar Problem some time ago when implementing a cache of (rather large) objects, but didn't want to make copies of every object when iterating over them (i.e. for sorting or filtering purposes). My solution was to have a generic function "Iterate" which is instantiated with the cache and a function "operation" that will be doing the work on the elements. The elements are passed to operation (in this case) as 'in' parameters, so 'operation' will not be able to change the elements, only read them.

I wonder whether container implementations couldn't use a similar trick: Instead of dereferencing a "do_with" generic would take an operation as a parameter to which in turn the elements are passed (control passes to do_with, do_with dereferences and passes elements to operation).

This doesn't avoid the problem of cursors that become invalid. The solution I see there would be to avoid cursors altogether and use generic traversal and iteration functors like fold() for lists. This way position is never made explicit and cannot be saved into a variable.

What about a tagged type that is the abstraction of a storage cell with method like Get(), Set() and Pass_Me_the_Data()?

I've not thought much about the details, but that should cover the most common use cases.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Amount of copying on returned constant objects
Newsgroups: comp.lang.ada
Date: Wed, 20 Jun 2007 09:34:06 +0200

Actually it would be only a half of the solution. The access type for which .all gets overridden is an implementation of the cursor. The public view of, should be a derived type of Element:

```
-- This is far not Ada!
type Cursor is new Element with private;
-- I can do with a Cursor anything I could
      do with an Element
function Get (Collection : Container,
      What : Key) return Cursor;
-- Factory
private
  type Cursor is access Element with
        interface Element;
   -- Privately Cursor is an access type
        that implements the interface
   -- of Element. It could be also a fat
        record rather than access, with
   -- a reference to the container and an
        index within. Whatever, it
   -- cannot be seen from outside.
function Is_Nice (Item : Cursor)
        return Boolean;
-- Implementation of Element's interface
        by Cursor
--  + Getter/Setter, of course
```

Here the referential nature of Cursor is hidden and Cursor is a full substitute of Element

Generation of optimized machine code

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: sub-optimal code for packed boolean arrays — bug or inherent limitation
Date: Mon, 9 Jul 2007 21:06:34 −0500

> Some inline assembly is unavoidable. There are two other bitboard operations that are required: First_One(Bitboard_T) returns Natural and Last_One(Bitboard_T) returns Natural; They return the position of the least significant and most significant set bit respectively. On x86 processors you can do this with one instruction BSF (bit scan forward) and BSR (bit scan reverse), and you cannot expect a compiler to generate them.

There's a reason for that: Intel recommended to compiler writers to not use "complex" instructions, as sets of simple instructions are often supposed to be faster. The "complex" instructions were implemented essentially as "macros" of simple instructions. (This is information is somewhat old, so it may vary on the most recent processors.)

My point is that it might not actually be faster to use those instructions than to use a loop of your own design (and there is a small chance that they'd be slower). As suggested elsewhere, you need to test that in a suitable benchmark. The number of instructions has had no real bearing on the execution time since the 486 came out (and indeed, given that some instructions are much slower than others, it never had much bearing on Intel's x86 processors).

From: Steve <steved94@comcast.net>
Newsgroups: comp.lang.ada
Subject: Re: sub-optimal code for packed boolean arrays -- bug or inherent limitation
Date: Tue, 3 Jul 2007 20:22:36 −0700

> As for the proof that the hand coded version is faster, here it is:

> Hand coded: 2 MOV, 1 SHL, 1 XOR

> Compiler: 2 MOV, 1 SHL, 1 XOR, 1 SHR, 1 ROL, 2 AND, 1 OR

> 5 extraneous instructions, Q.E.D.

Twenty years ago I might ave agreed with this logic, but not today.

In the good ole' days you could associate an amount of time with each instructions, add them up and get a total amount of execution time. This hasn't been possible for a long time. Ever heard of "instruction scheduling" and "concurrent execution"?

Today's CPUs contain "pipelines" that sometimes merge several operations into a single clock. In some cases you will find that NOPs are added to increase the speed of execution based on a detailed knowledge of the underlying processor.

I agree with Jeff's assessment. You must benchmark to measure and compare performance.

I work frequently with time intensive code where simulating more permutations translate to more value recovered. These small differences in code generation seldom have a significant impact on the overall performance. Greater benefits are found by changes to algorithms or approaches to a problem.

From: Robert A Duff <duff@adacore.com>
Newsgroups: comp.lang.ada
Subject: Re: sub-optimal code for packed boolean arrays -- bug or inherent limitation
Date: Sun, 08 Jul 2007 18:53:11 −0400

> You should also be aware that you're sending the compiler conflicting messages. Packing the array indicates

> that you want the compiler to minimize storage, even at the expense of speed, …

Well, sort of. Pack means to minimize storage even at the (possible) expense of speed of accessing the components. Speed overall can be improved by packing. For example, speed of copying and comparing the whole packed object, and passing it around as parameters, will typically be improved by packing.

And of course using less memory will typically improve cache and paging performance.

So on a desktop computer with "plenty" of memory, packing is mainly used to improve speed — there's no "ime/space" tradeoff. A small embedded system might have such tradeoffs.

> … while "pragma Optimize (Time);" indicates that you want to minimize time, even at the expense of storage. Obtaining speed often requires wasting storage; the fastest

I'd say, "sometimes", nor "often". Usually, wasting storage means wasting time.

> implementation might be to not pack the array. I don't know how many of these you have, but with the large memories on modern machines, you might be better off with the additional 56 bytes per array. Again, you won't know for sure until you have a working version you can measure.

Measurement is a Good Thing.

From: Jeffrey R. Carter <jrcarter@acm.org>
Newsgroups: comp.lang.ada
Subject: Re: sub-optimal code for packed boolean arrays -- bug or inherent limitation
Date: Tue, 03 Jul 2007 18:36:31 GMT

> Now, concerning efficiency: these basic operations on bitboards are used 1e8 to 1e9 times every time the program tries to decide what to move next. Even the smallest improvement in speed can mean the difference between searching 10 or 11 ply deep, which can mean an improvement of over 50 Elo points in strength.

I'm sure that's true. However, you have not demonstrated that there's a speed difference between the 2 versions. Even if there is, it doesn't necessarily mean that there will be a difference in the # of plys that can be searched. The old rule, "1st get it right, then make it fast," still applies. Once you have it finished you can easily see if modifying this single procedure actually makes a difference. Until then, you're wasting a lot of effort on something you don't know is important. […]

From: Jeffrey Creem <jeff@thecreems.com>
Date: Wed, 04 Jul 2007 06:15:19 −0400
Newsgroups: comp.lang.ada
Subject: Re: sub-optimal code for packed boolean arrays -- bug or inherent limitation

> My reaction is due to an allergy to premature optimization, the root of all evil.

"Premature Optimization" is one of those pieces of conventional wisdom that gets repeated often but which is in fact not the problem at all. Not really.

In any real-world problem, you rarely get to spend 6 months making it faster because you did such a poor job up front in designing the problem.

Often, by the time one has created a design mess that is slow everywhere, spending the short time optimizing the scary loops is not enough because the design itself is broken.

The real problem is not so much premature optimization but crazy micro-optimizations done early that also hurt maintainability/clarity of the code and often do little if anything to actually make the code faster.

Spending time up front thinking about the overall design and even worrying a little about specific performance details when not taken to the extreme and when done with the benefit of metrics backed experience (v.s. I heard one time that compiler X was slow on this) is actually not all that bad.

This is of course especially true when one considers that by the end of the project when you go to start profiling the code to find the hotspots and find out that your lousy tool set does not support profiling with programs with tasks, or does not support profiling of non-trivial programs at all.

AVR-Ada quality

From: Simon Clubley <clubley@eisner.decus.org>
Newsgroups: comp.lang.ada
Subject: AVR GCC/GNAT port quality ?
Date: 25 Jun 2007 06:47:35 −0500

I'm thinking of using the AVR microcontroller along with the AVR GNAT compiler in a project.

Would anyone here like to offer an opinion on the quality of the GNAT AVR port ?

From: Rolf Ebert <rolf.ebert@gmx.net>
Newsgroups: comp.lang.ada
Subject: Re: AVR GCC/GNAT port quality ?
Date: Tue, 26 Jun 2007 12:04:10 −0700

Nice to see that someone is interested in AVR-Ada. I can only encourage you to try it out. I built a data logger (several temperatures and gas consumption) for my home based on the cheap AVR-Butterfly. And I also managed to make the first steps with the Asuro robot [1]

The Ada compiler itself is quite good. When I started the project some years ago I had some ICE (internal compiler errors); they all seem to be ironed out by the time.

But then you have to be careful what Ada constructs you want to use. Most of them require extensive support in the run time system. For most constructs I do not yet provide the necessary files in the RTS. Most notably there is no support (yet) for:

- tasking
- exceptions
- run time dispatching (aka tagged types)

On the other hand you can:

- access all IO pins
- attach procedures to interrupts
- use some predefined routines for LCD, RS232, flash memory, eeprom, etc.
- have a lot of fun squeezing Ada programs into a few hundred bytes.

I recommend you to join the mailing list at http://lists.sourceforge.net/mailman/listinfo/avr-ada-devel or send me some direct email.

Trampolines vs. Thunks

From: Jerry <lanceboyle@qwest.net>
Subject: LLVM--Low Level Virtual Machine--and Ada
Date: Tue, 17 Jul 2007 20:56:35 −0700
Newsgroups: comp.lang.ada

I've been hearing a lot lately about LLVM, the Low Level Virtual Machine. Apparently it is kind of like the GCC arrangement with a front-end and back-end for compilers. The difference with the LLVM is that it is supposed "super easy" to make a front end. So naturally one wonders, what is the likelihood of Ada being supported?

http://llvm.org/

From: Duncan Sands <baldrick@free.fr>
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Wed, 18 Jul 2007 09:48:39 +0200
Newsgroups: comp.lang.ada

I'm porting GNAT to it. Most likely it will not be in the next LLVM release, but in the one after that. However it should be possible to check out a development version in the near future. I will post an announcement here once something usable is publically available. It works quite well in my development tree, for example all the ACATS tests pass except for a bunch of tasking tests (I don't know why those fail yet).

From: Duncan Sands <baldrick@free.fr>
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Thu, 19 Jul 2007 16:56:26 +0200
Newsgroups: comp.lang.ada

> I read LLVM specification long time ago. There weren't many operations

> needed to run Ada programs on it. Such as overflow checks on integer arithmetic (even divide by 0),

These are generated as explicit conditional statements by the front-end, the same as for gcc, so no special LLVM support needed here. If it had special support that would be great of course. But it is no worse than gcc in this respect.

> variable access from nested subprogram by lexical level,

I've implemented nested functions and (for the Intel x86) pointers to nested functions.

> allocation in stack variables of unknown (till runtime) size,

All these kinds of variable size things have been implemented.

> multitasking,

These is done using library calls, so no special support is needed.

> asynchronous jumps,

The only such jumps are for exception handling. I recently helped complete the LLVM exception handling implementation. […]

From: Robert A Duff <duff@adacore.com>
Newsgroups: comp.lang.ada
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Thu, 19 Jul 2007 17:30:07 −0400

> PS: the only thing that needed new LLVM functionality, i.e. functionality that didn't exist and wasn't being worked on, was pointers to nested functions.

How did you implement those?

GNAT uses trampolines, which are less than ideal for several reasons.

From: Duncan Sands <baldrick@free.fr>
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Fri, 20 Jul 2007 09:44:06 +0200
Newsgroups: comp.lang.ada

I used trampolines. I really couldn't do it any other way without a bunch of modifications to the GNAT front-end. Also, this seems to be the only solution if you want to be able to pass such pointers to foreign language routines, which seems nice to have if not essential. The major downside I see is that they seem to be extremely expensive, presumably because you take an icache hit every time you jump to the stack. I plan to implement a bunch of small optimizations which may help, such as converting direct calls to trampolines into direct calls to the nested functions (inlining hopefully will create such direct calls). Any suggestions for a better approach than trampolines? I'm also curious to hear what the other less than ideal properties of trampolines are!

From: Robert A Duff <duff@adacore.com>
Newsgroups: comp.lang.ada
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Fri, 20 Jul 2007 10:06:25 −0400

The main problem with trampolines is what you said — they're slow.

Another problem is that some modern machines use DEP (which I think stands for "data execution prevention" or something like that). DEP means the operating system prevents writable data from being executed as code. The purpose is to prevent certain kinds of security holes that are common in languages that don't do array-bounds checking. But DEP prevents trampolines from working, so users have to turn it off in order to run some Ada programs (such as the compiler). It's a pain because users get some mysterious error message when trampolines are used. […]

The alternative to trampolines is to represent access-to-nested-subprogram as a pair, sometimes called a "fat pointer": (address-of-code, static link). AdaCore is thinking about doing this at some point. You are correct that this won't work when interfacing to C. The answer is: if the Convention of the access type is C, and the procedure is nested, use a trampoline. Trampolines are probably also required for 'Unrestricted_Access, because it allows you to bypass the normal accessibility rules.

Access-to-nested subprograms is much more important in Ada 2005 than in Ada 95, because we now have downward closures. (See the Iterate procedures in the Containers packages for examples.) When using downward closures, you almost always want nested subprograms. (Downward closures are one of my favorite features, by the way.)

Another issue is nested type extensions — dispatch tables contain pointers to subprograms. E.g. type T1 is library level, and "T2 is new T1" is nested. The primitives of T2 need a static link. You don't want to store static links for T1 — that would be a distributed overhead. T2's primitives can be trampolines, but that's inefficient. There's a trick mentioned in the AI that introduced this feature: store the static link as an implicit component of T2. T2's primitives can be a wrapper that loads the static link and then calls the user-defined code. This eliminates any overhead on T1, and is still more efficient than trampolines for T2. See the AI for details.

One use for nested type extensions is when you want a Finalize procedure of a controlled type to be nested.

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Mon, 23 Jul 2007 21:12:08 −0500

> I'm not sure that [DEP] is a problem anymore: gcc uses a bunch of tricks (eg: setting a flag on the program that notes it runs code on the stack) to inform the operating system that the trampoline is kosher IIRC. That said, I haven't tried to implement any of this in LLVM yet, which is also why I'm vague on the details.

That would be bad, as it would effectively turn off DEP for LLVM programs. These error detections are critically needed and turning them off just means you have buggy software that you can't/won't fix and that you're willing to remain part of the problem.
Honestly, I never understood why programs *ever* needed to execute permission on stack and data. When we did our first 32-bit compilers, I kept those segments completely separate and was dismayed to find out that we couldn't set the permissions on the segments to actually match the uses (and thus detect bugs earlier). I managed to get the DOS extender versions sort-of-right by discarding the overlapping writable segments given to us by the OS and creating new non-overlapping ones for the data and stack. But neither Unix nor Windows provided anything that helped at all. I find it bizarre to find people deciding to apply the obvious technique of least privilege nearly 20 years later — what the heck have they been doing in the mean time? (Not caring if software is correct is one obvious answer…)

Janus/Ada has never used any executable data/stack in its 32-bit versions; such code would save no more than a clock cycle or two (out of hundreds or thousands) and as such could not be significant. We use compiler-generated thunks rather than run-time generated trampolines, and I'm not sure why anyone would use the latter (given that they increase the exploitability of a program). Must be something I don't understand…

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Tue, 24 Jul 2007 14:21:30 −0500

> How do these thunks work?

Generally, the compiler passes the address of the thunk to wherever it is needed, and it is called indirectly. The thunks themselves adjust the parameters as needed and call the real routine.

This is necessarily a very general description; we use many different kinds of thunks, and the details are different for each. There are some where the address of the routine to call is passed in as well; some of them are just wrappers, and some implement entire operations (like an allocator).

In general, the front end generates the thunks; the back end knows about them for optimization and debugger and error message purposes, but it doesn't generate any. I would expect it to be fairly hard to retrofit them if the front end is insisting on doing something else.

(It's also possible that trampolines are much faster on some architectures; I've primarily looked at the x86 machines where there is little advantage.)

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Tue, 24 Jul 2007 14:58:03 −0500

> Presumably pointers to nested functions are "fat pointers" containing both the identity of the nested function (equivalently, the identity of a "thunk" that knows about the nested function) and also some kind of pointer into the stack of the parent of the nested function. Is that right?

Janus/Ada uses displays rather than static links (that was originally a requirement of the University project that led to Janus/Ada, and we never changed it), so in most cases, you don't need any special stack pointers. (You only need them when you call to a place that is outside of the normal nesting of subprograms.) There are some cases when you do need them, and in those cases we need to provide a replacement display. That is usually saved in a well-known place (i.e., for shared generics it is part of the generic data block) and shared with a number of related thunks, although for anonymous-access-to-subprogram it will be part of the pointer. Even in these cases, we determine in the thunk how much of the display needs to be replaced (it is often none) so as to keep the overhead to a minimum, and all of this code is part of the thunk, not part of the call site.

From: Robert A Duff <duff@adacore.com>
Newsgroups: comp.lang.ada
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Tue, 24 Jul 2007 19:28:30 −0400

Are you saying that executable data necessarily means the program is buggy? If so, I don't agree. Trampolines are not bugs. They're slow. Turning off DEP might expose _other_ bugs that cause security holes, but those can be detected/prevented by array-bounds checking and the like.

The need for DEP is really because we live in a C world. And DEP doesn't even solve the problem.

There are several legitimate reasons why a program might want to execute data. For example, consider a JIT compiler.

I agree with you about "least privilege". To me, that means that writeable stack/data areas should be no-execute by default. But a program should be allowed to change that. […]

Here's the reason for trampolines:

The GCC dialect of C allows nested functions (unlike standard C).
They wanted to allow pointers to these functions.
They wanted to make those pointers fit in one machine word, for two reasons:
Some C programs might assume a pointer-to-function has that representation (a single address — of code).
Programs that obey the standard (i.e. do not nest functions) should not pay extra (no distributed overhead). […]

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Tue, 24 Jul 2007 19:39:26 −0500

The vast majority of programs have no need for executable data. Which includes virtually all Ada programs, and it is bad to be turning off things which protects the program from bugs.

Of course it doesn't "solve" the problem, but it surely helps. And remember that even Ada programs have to call C code to access OS facilities and the like. Probably 50% of the bugs (and by far the worst ones) are in those interfaces, where the checking of Ada is no help. (And another 10% is compiler bugs that would have been detected by DEP; overwriting the return address of something is probably the most common symptom of compiler bugs that I have to track down. That may be because those are the hardest to find…)

How many people are writing JIT compilers or overlay managers so they can get 2 megabytes of compiler code to execute on a 640K MS-DOS machine?? Hardly anyone, and people who need to do such things should be put through extreme hoops before being allowed to do so. (I'd suggest that they be required to write all of the JIT code in SPARC or a similar proof system, except that SPARC won't allow such dynamic things — it's major failing IMHO.)

> A pointer-to-nested function needs access to the function's environment, and the only way to do that, while keeping such a pointer in a single word, is trampolines.

That's not quite true, as the use of displays rather than static links would allow up-level access without any fat pointers. Although they might have wanted non-nested access, for which that wouldn't work. (Non-nested access shouldn't work IMHO, because of the significant risk that the stack frames no longer exist when the routine is called. That's the reason the anonymous access-to-subprogram parameters can't be converted to other kinds of access-to-subprogram.)

In any case, Ada has restrictions such that these aren't needed for most access-to-subprograms, and fat pointers are fine for anonymous access-to-subprogram parameters (because you can't give these a convention and thus they don't need to be interoperable with C or anything else). GNAT is probably screwed though, because of the mistake of 'Unrestricted_Access for subprograms. Still, I think GNAT should do what it can to avoid them; most Ada access-to-subprogram types don't need them (although you would have to insist on having convention C on those types that are intended for interface use; perhaps that would be unacceptable).

> Trampolines are not efficient! Because after writing the code, the instruction cache needs to be flushed before calling the thing — that is expensive on modern machines.

True. Thunks have similar issues in that the address of the thing isn't known before the call (so there is a pipeline stall), but at least we don't have to flush the cache. And there sometimes is the possibility of in-lining the call (something we don't currently do, but it is on the radar).

From: Robert A Duff <duff@adacore.com>
Newsgroups: comp.lang.ada
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Tue, 24 Jul 2007 22:00:59 −0400

[…] Ada itself does not require trampolines.

So why use them? Well, I suppose it was an easy implementation. They could be got rid of, except in the cases of interface-to-C, and 'Unrestricted_Access.

[…] That's how it works in gcc C -- you can pass a pointer to a nested function to another procedure, and that thing might save it in a global, and then call it later. It works, so long as the outer function is still active. I.e. it's unsafe. Ada prevents that sort of thing.

'Unrestricted_Access might need to use trampolines. The standard Ada features can avoid them. […]

From: Duncan Sands <baldrick@free.fr>
Subject: Re: LLVM--Low Level Virtual Machine--and Ada
Date: Wed, 25 Jul 2007 15:14:20 +0200
Newsgroups: comp.lang.ada

> […] Trampolines are worse than thunks, efficiency-wise.

> But it's only in Ada 2005 that it started to matter much.

In programs that use Ada.Containers heavily, my profiling tools regularly show "code on the stack" (i.e. trampolines) as where the program is spending most of its time.

Emmett Paige's 1997 DoD Memo

From: Richard Riehle
<adaworks@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Sun, 26 Aug 2007 10:51:38 −0700

[…] When Emmett Paige wrote his famous memo abrogating the Ada mandate, the memo was widely interpreted as the equivalent of the DoD admitting that Ada was a mistake, and direct abandonment of its use for future DoD projects. Although that was not the intent of the memo, that interpretation is now widespread both within and outside the DoD.

It is unfortunate that the memo was written in a way that left it open to Ada's enemies to misinterpret. The damage done is widespread. The educational institution where I teach once required Ada of its students. Now the language is almost non-existent except in a two-week portion of an eleven week class that I teach. No one else in our computer science department gives it any credibility at all.

The real-time software projects are now being written in Java. The funding for research will not support anything with the Ada language involved. The newly-hired faculty members regard Ada as a quaint era of the past, not something to be taken seriously.

I have been an Ada advocate for about twenty years, but it is becoming clear that, without some miracle or absent someone in the DoD coming to their senses, the use of the language will continue to decline both in the commercial world and in the DoD. When I was still consulting and teaching Ada, one of my major clients, a DoD contractor building one of our major weapons systems, switched from Ada to C++. It was a massively stupid decision. But the man who was previously in charge, who understood the value of Ada, retired. His successor knew little about Ada and was a strong advocate of C++. Without the mandate in place, he could blithely ignore the wisdom of using Ada and demand that everything be written in C++.

I asked the question, at the time, "What makes you think you can use a language such as C++ that is inherently error-prone, and expect a result that is error-free." My credibility suffered from my resistance to C++. The more I saw of, and continue to see of, C++, the more I realize how dangerous the language is and how wrong-headed it is to use C++ for military software systems, but my opinion carries no weight. At the same time, in an effort to offset the known dangers of C++, many DoD organizations and their contractors have chosen Java. This is also a dumb decision, but the new real-time features of Java make it more difficult to clarify the points that make Ada a better choice.

There is no single strong advocate for Ada at present. There is no powerful corporate sponsor as there is for Java. There is no major Ada project that is visible to the larger community of software developers. The language is seen as "old-fashioned" and out-of-date by those who have graduated within that past ten to fifteen years. It is an oddity.

The damage to Ada was the result of many factors. The AJPO never quite got it right. The DoD certainly never got it right. The infighting between Ada vendors never helped. The fact that Ada compiler vendors charged outrageous prices for their compilers helped to discourage commercial organizations from using Ada: COBOL, C, C, Pascal, were more affordable. Most PC versions of Ada had less capability for building PC applications directly than BASIC. With exception of the Meridian Compiler, there were no good libraries for creating MS-DOS applications. Even Meridian got it wrong by defining the data type for system address incorrectly.

With Ada 95, the designers and contributors to the design of the language did get a lot of things right. Ada finally became a language for the ordinary programmer. The time was also right. A lot of people renewed their interest in the language. Then, grabbing defeat from the jaws of potential victory, the letter from Mr. Paige muddled the entire decision-making process. A delay of two or three years before writing that kind of letter might have made a difference. Instead, the developer community ran as fast as it could to find other options.

JSF is being developed in C++. A truly dumb decision. Missile Defense Agency has completely abandoned Ada. […]

This is truly unfortunate. Ada continues to be the best hope as a language for software engineering. In my view, it is still the best language for use in safety-critical, mission-critical, and military software systems. It offers a lot to commercial software developers, as well. How we get that message out, now that there is no powerful sponsor and no effective Ada consortium, I don't know. At one time, I used to write a lot of articles about the value of Ada for software magazines such as JOOP, HP Professional, Embedded Systems Programming, and others. That seemed to help a little. I have yet to see anyone publish an article about the Ada 2005 standard — even in DoD publications. It is as if it never happened.

I no longer have the time to devote to Ada since my role has changed. I am no longer directly involved in Ada, though I continue to promote it whenever I can. I can still teach it in some of my classes, but I get the question from my colleagues, "Why are we bothering with that old language?" At present, I am the last hold-out for keeping Ada in some small part of our curriculum. When I am gone, Ada will also be gone. Or as newer faculty members take over my courses, Ada will vanish entirely.

I wish I could outline an action plan instead of posting a tale of lament. Perhaps someone from this forum can come up with a solution for improving the situation. I wonder if someone might write and publish some articles about the new standard and the continuing viability of the language? Maybe we can get someone in the DoD, someone with a brain in their head who understands software, to reinvigorate and reinstate the interest and committment to Ada. I would hope so, but it is a faint hope at this point.

From: Gary Scott
<garylscott@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Mon, 03 Sep 2007 11:36:10 −0500

[…] In one particular environment (test equipment), all of themodels, real-time data capture/processing, etc. were in extended Fortran 77 plus embedded and standalone assembly modules. It wasn't that C would have improved the specific product at all, it was very well structured (although non-portable, but it was very hardware specific so it wouldn't be portable in any language). It was the feeling that the world was passing them by as EVERYTHING was in Fortran 77.

At one point, the directive to use Ada applied to this environment as well so they began porting to Ada. However, the Ada compiler was so new and inefficient (little optimization), the application set would no longer execute on a system with several times the memory and CPU capacity of the Fortran/assembly based one. It eventually was completed, but this experience negatively tainted management against Ada. No other attempts were ever made that I am aware of to use Ada for the test environments. Likewise, there was no concerted attempt to understand WHY the Ada development foundered. It was of course a mixture of operating system inefficiency, compiler inefficiency, and software/hardware architecture inefficiency.

The older system used extensive proprietary parallel processing, DMA, and shared memory and the new system used COTS message passing schemes. Before the advent of fast CPUs, there simply was no other way to accomplish the task in a cost efficient manner than to use parallel processing. With the advent of fast CPUs, much less thought goes into the hardware design with the thought that the CPU is so fast, we'll just emulate that part

of the hardware in software or perform its processing job in a separate process or thread without really thinking through the overhead (cache utilization, interrupt processing, task switching time).

From: Gary Scott <garylscott@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Sun, 02 Sep 2007 15:03:54 −0500

[…] I have discussed these issues with many programmers and it is a somewhat pervasive attitude that not keeping their C language skills honed places them at a competitive disadvantage. Defense has somewhat frequent employment ups and downs. They simply want to be competitive with those competing for commercial jobs.

I had a conversation with Nancy Leveson (Safety Critical Software). She tends to be somewhat language agnostic in her books, but it is my belief that she agrees with the above but is hesitant to voice such a heretical view.

From: Richard Riehle <adaworks@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Wed, 29 Aug 2007 05:31:37 GMT

Today I presented the Ada module in my "programming paradigms" course. The reaction from students is good. Soon, though, I may lose that course to one of our junior faculty who knows nothing about Ada.

Ed Falis indicated surprise that anyone could misinterpret Paige's memo. The fact is that the misinterpretation was widespread and that wrong interpretation was received with a certain amount of glee in some quarters. For some reason, Ada has enemies. It is not entirely clear why this should be the case.

From: Ed Falis <falis@verizon.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Wed, 29 Aug 2007 14:27:29 GMT

Richard, you misinterpreted the sense of my comment: I don't see how anyone could have interpreted Paige's memo as anything other than DoD walking away from Ada.

From: Richard Riehle <adaworks@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Fri, 31 Aug 2007 07:25:08 −0700

> Which does not contradict my statement in the context of the times. Despite the superficially "fair" wording of the memo, it was almost universally interpreted as DoD walking away from Ada. One of my colleagues was at a meeting recently where some yoyo got up and said "Thank God we got rid of Ada"! Probably because that was the "cool" view among those who felt oppressed by the mandate (largely in terms of their short-term profit margins).

I published an article in Crosstalk several years ago that attempted to clarify Mr. Paige's intent. I even sent him a draft of the article for his approval before publishing it. He agreed with my assessment and the content of the article.

His original hope was that, having been proven successful in a lot of DoD projects, Ada would stand on its own and be chosen without the coercion of a DoD mandate. It has been suggested by some that there was a lot of "behind the scenes" influence from DoD contractor executives to get rid of the Ada mandate. There may have been some of this, but there was also a lot of controversy generated in other quarters.

Some people in this forum may recall the flurry of email and forum postings from some pipsqueak (I cannot recall his name) who constantly bombarded Mr. Paige and other DoD executives with diatribes about both Ada and their management of Ada. It did not help at all that some former AJPO officials, in particular Don Reifer, became turncoats and used their visibility in the software industry to publicly denigrate and discourage the use of Ada in DoD publications such as Crosstalk.

I used to do a lot of training and consulting for Lockheed and CSC related to the Aegis project. Soon after the Paige memo, Lockheed dictated that the software for Aegis would be written in C++ instead of Ada. Almost all training in Ada stopped, and the programmers were given intensive training in C++. I told everyone that it was a big mistake, but my advice was of little interest to those who were already biased toward C++. The answer was, "We can find C++ programmers right out of university CS programs, but no one teaches Ada in CS."

A lot of the early frustration with Ada 83 was justified. There were things one could not do easily with it. Some of the work-arounds required on some projects were horrible. There was no language defined data type for unsigned integers and I recall a project where that took a lot of time away from the programming effort just to invent a work-around. Hobbyists, many of whom were more influential than anyone realized, found they could not easily format a simple MS-DOS screen with most compilers. The compiler vendors resorted to ANSI.SYS, which was simply another work-around. Alsys did have a special package that supported an unsigned integer, and I recall a USMC project where we were able to access B800(Hex) area of memory to directly access the video display mapping.

With Ada 95, a lot of things got better. We no longer had to make excuses for, nor invent work-arounds for, that lack of inheritance. It does not matter who made the mistake of excluding inheritance from the language in the first place. I remember many discussions where I was defending Ada 83 because it did not support extensible inheritance. As it turns out, we still don't use inheritance that much for safety-critical software anyway. And we certainly don't use dynamic binding.

In spite of the good efforts of people like Ed Falis and Ben Brosgol at Alsys, commercial adoption was a failure. In fact, it was due to the efforts of those two people that Ada 95 did become hospitable to commercial and business data processing applications. Unfortunately, the compiler publishers ensured that no one in the commercial world would use Ada by: 1) pricing the compilers so no one could afford them, and 2) separating Ada from the rest of their product line by relegating it to a sales option for their Federal division. At IBM and Rational, very few people on the commercial side of the sales force had any knowledge of Ada.

The consortiums (ARA, etc.) found a way to waste money on some of the most absurd ad campaigns ever launched. Does anyone remember those ridiculous ads in the late 1990s. That was money down the drain.

Ada continues to be the best option for safety-critical and military weapon systems. I work in a DoD organization and try to promote it whenever I can. My reasons for promoting Ada for DoD software have little to do with Ada, per se, but with my concern about the dependability of software that must work right every time it is used. With Ada we have a better chance of achieving that goal than we do with C or C++, or even Java. I have even been called an "Ada bigot," and sometimes described as a "throwback" for my views on programming language choice.

As nearly as I can tell, my continued advocacy of Ada for DoD software puts me in a very small minority of the "quaint but tolerated" software community. Most of my Ada-knowledgeable colleagues have given up the fight and gone on to other things. They have concluded that C++ is good enough; Java is good enough; Python is good enough. One of my students told me recently of a flight-control system on one of our military aircraft where the software is written in VisualBasic. I hope he is wrong.

When the Paige memo came out, I commented in a public article (in JOOP) that, if the DoD cannot manage a single language policy, how do they expect to manage a multiple-language policy. They

can't. They have decided to let the contractors make the choice. The long-term consequences of this abrogation of responsibility will be dire, but no one seems to care.

I realize that many in this forum are not concerned with warfighting software. Perhaps the commercial software you are developing will make enough difference that some of those in the DoD who need to understand the issues of software decision-making will come to their senses when they see the results of your work. However, it is too late for influencing the DoD contractors. They are now free to use any language they wish, including some proprietary language they might invent or extensions to some existing language that no one else knows about.

The Paige memo did its damage. Now we need to find some way to repair that damage. It might be too late. On the bright side, SPARK is "sparking" renewed interest in Ada — as long as we don't call it Ada.

From: Adam Beneschan <adam@irvine.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Fri, 31 Aug 2007 10:18:42 −0700

[…] I frankly wouldn't expect good results from *anyone* who can program in language X because it's what they learned in college but couldn't pick up language Y; to me, I wouldn't trust someone like that to have a real understanding of "software" or "programming", and because of that I wouldn't expect them to write good software no matter how good language X is, even if it were Ada. Ada is not a good enough language to make up for a fundamental lack of software engineering understanding. […]

From: Ole-Hjalmar Kristensen <ole-hjalmar.kristensen@sun.com>
Organization: Sun Microsystems
Date: 04 Sep 2007 09:07:12 +0200
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?

[…] In the late eighties I worked on an automated toll gate system, and among the team were two junior members. One had EE background and lots of experience with C. The other had CS background and no experience with C whatsoever, but a thorough understanding of software engineering. After a couple of weeks on the project the CS guy was definitely more productive in terms of delivering code that worked….

From: Richard Riehle <adaworks@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Sun, 2 Sep 2007 12:04:57 −0700

> But part of the issue has been unhappiness of the programmers themselves. When told that they would have to program in Ada, the C programmers were turning down job offers. Not because they couldn't pick up Ada, but because they wanted to keep their C skills polished in case they found a better position elsewhere. You do get rusty from non-use, and you fall behind the latest standards over time.

I have heard this argument from the so-called managers who were using it as an excuse for not using Ada. When the interviewing manager says something such as, "Of course, in our shop you will be programming in Ada instead of C. I know this is a little bit out of the mainstream, but the government programming we do requires us to use Ada." Or some similar line of apologetic interviewing, what can we expect. Yes. Too often, the managers would apologize for using Ada instead of focusing on the benefits of using it. And there are a lot of benefits. Adam mentioned the software engineering benefits, and those benefits are substantial.

When I was just a programmer, even a programming manager, before discovering Ada, I did not really understand software engineering very well. Most of what passed for (and still passes for) software engineering was the adoption of Industrial Engineering protocols on the software process. There was very little of what any real engineer would call engineering. I have Ada to thank for helping me rise above the programming model that I had been stuck with for so many years.

Hardly anyone engineers software in C. Very few really use C++ to engineer software solutions. As long as we remain tied to the notion that programmers are the driving force in the software process, we are doomed to a long nightmare of horrible applications where debugging is the norm and design is the exception. If C++ is the answer, someone is asking the wrong question.

Where C is often called a "universal assembler," C++ is an object-oriented assembler, and not as universal as C. If software engineering is, in part, about levels of abstraction, C++ is at a very low level of abstraction. As long as we continue to think of software in terms of computers instead of in terms of the required solutions, we will be stuck with a model of software that continues to focus on the low-level issues.

When I first began to learn Ada, coming to it as an old-fashioned programmer, it was a strange and difficult transition. My first inclination was to look for ways I could leverage Chapter 13 for my code. It took a while to understand the finer points of the language. Once I was able to understand those, it seemed strange to me that I used to write programs in a different way.

Sadly, those LMCO managers on Aegis who made the decision for C++ instead of Ada simply don't understand Ada. They are still thinking in terms of programming languages, not in terms of engineered software. This is true of most of the DoD contractors I have known over the past twenty+ years. They have no idea of the benefits of software engineering, something they can do with Ada better than with most other options. It is a matter of ignorance, nothing more. If they did understand the difference, there would never have been abandonment of Ada in favor of C++.

So, instead of learning how to apply good software engineering principles, most of them have behaved like human lemmings, blindly following the idiotic choices made by those in the software industry who also know little about engineering, but a lot about programming.

Until the DoD, and industry in general, begins to take more of an engineering approach to the development of software, we will continue to wrestle in our bedclothes with the software nightmares that continue to haunt us, only to wake in the morning and discover that our best efforts to control those nightmares have consummated themselves in nothing more than a simple wet-dream.

Multithreaded callbacks

From: Maciej Sobczak <maciej@msobczak.com>
Newsgroups: comp.lang.ada
Subject: Interfacing to C: multithreaded callbacks
Date: Tue, 12 Jun 2007 12:56:30 −0700

I have identified three problems with interfacing to C, I will describe them in separate posts.

Suppose there is a C library that creates additional threads (system- level threads in the pthread_create sense) and can call the client code back via function pointers that the client code provides to the library. Asynchronous I/O library that notifies the client about state changes can be a good motivating example.

It is possible to pass Ada callback to the C library — it's enough to pragma Export(C, My_Procedure) and pass appropriate access to procedure. This way we could, for example, use the standard C function qsort. The problem is when the C library creates additional threads and calls the client back in the context of those threads. ARM says nothing (?) about the relation between Ada tasks and system threads. If the relation is 1:1 (ie. tasks are implemented as system threads), then the whole scheme might work just fine, provided that there is no task- specific data that Ada runtime expects and will not find. On the other hand, if the relation between tasks and threads is not 1:1, we

will just enjoy undefined behavior. Looks like a shaky ground.

Is there any water-proof implementation pattern for such problems? Consider both the general case and then GNAT as the target Ada compiler on POSIX systems.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Interfacing to C: multithreaded callbacks
Newsgroups: comp.lang.ada
Date: Wed, 13 Jun 2007 10:11:30 +0200

Yes, when Ada tasks aren't mapped onto system threads, then system calls potentially might block all Ada tasks. This includes whatever POSIX layer.

You can marshal messages from C callbacks. With busy waiting and one publisher — one subscriber, you don't need anything but shared memory to implement that.

However, when talking about POSIX targets, I would assume Ada tasks being POSIX threads.

From: Maciej Sobczak <maciej@msobczak.com>
Newsgroups: comp.lang.ada
Subject: Re: Interfacing to C: multithreaded callbacks
Date: Wed, 13 Jun 2007 08:23:08 −0700

> You can marshal messages from C callbacks.

Yes, that might be some possibility. Going further in this direction, I might even isolate the C library in a separate process and communicate with it from Ada client using some appropriate IPC. I can imagine scenarios where that makes sense.

> However, when talking about POSIX targets, I would assume Ada tasks being POSIX threads.

Yes, that should be obvious implementation strategy. It doesn't automagically solve all problems, though. The threads started by C library will be "pure virgin threads", without any Ada-related context information that might be stored in TLS (Thread Local Storage), for example. Crossing the border between C and Ada in a callback is a matter of calling convention and single pragma, but depending on what the Ada subprogram tries to do next it might work or not. Just imagine that such a subprogram will try to do some tasking-related stuff (rendezvous with other Ada task? etc.) and from the point of view of Ada runtime will be just a foreigner. I think that C threads should not pretend to be Ada tasks, unless we know *everything* about the particular Ada implementation. Some GNAT developers might shed some light here.

Fortunately, the C library I have in mind offers (possibly blocking) polling as alternative to callbacks, so that it should be possible to set up "normal" Ada task that will poll the library for state changes and then do regular Ada callbacks to other subprograms when needed. This way C threads will not mess around Ada runtime. But I can imagine C libraries that don't provide this opportunity; then marshaling or total isolation might be the correct solution.

From: Simon Wright <simon.j.wright@mac.com>
Newsgroups: comp.lang.ada
Subject: Re: Interfacing to C: multithreaded callbacks
Date: Wed, 13 Jun 2007 20:57:35 +0100

For GNAT, see GNAT.Threads.Register/Unregister_Thread — they seem to think it's quite tricky, maybe other vendors have a different slant.

Unknown binary layout interface

From: Lutz Donnerhacke <lutz@iks-jena.de>
Newsgroups: comp.lang.ada
Subject: Re: Interfacing to C: API with structures but without binary layout
Date: Tue, 12 Jun 2007 20:19:27 +0000 UTC
Organization: IKS GmbH Jena

> Another problem with interfacing: consider a C function that expects some data structure that is defined in terms of C struct definition, but without clear binary layout.

Hard problem even in C. The binary layout of passed structures might differ between different executables notably between kernel and application, if the C-compiler options of both are different enough.

> Now, in order to call such a function we need access to the appropriate header file, where the actual structure definition is provided. But Ada doesn't understand C headers and pragma Import(C, connect) will not be enough.

I wrote a very thin binding to the Linux kernel (resulting in a inlined INT 80 after all). If the API/ABI does not define the structure closely enough to use representation clauses, an Ada record with layout Pragma(C) did always the job.

> A straightforward solution is to create a thin wrapper (in C) for the given function that will expect parameters in well-defined form and layout and just forward them to the interesting system call. This wrapper will have to be compiled separately on each target system, picking the actual structure definition from the appropriate system header. Ada can then call the wrapper function.

> Is this THE solution?

It is the canonical way, yes. Given the majority of existing software, you can collect the information directly and provide platform specific Ada sources. This is causes by the platform specific API, which can't be described portable by a higher level language.

From: Jeffrey Creem <jeff@thecreems.com>
Date: Tue, 12 Jun 2007 18:36:16 −0400
Newsgroups: comp.lang.ada
Subject: Re: Interfacing to C: API with structures but without binary layout

There are a few possibilities.

1) Define the Ada records to match what you expect and hope for the best (not great but fine in some cases)

2) Create the wrappers that you have suggested (I've used this in some cases)

3) Use an automated bindings generator for some of the thin stuff and create the bindings during the ./configure step. This is certainly what some other languages do in some cases (e.g. Python). SWIG is a tool that can in theory be used in this cases. The current official sources do not support Ada but the maintains have indicated they will accept patches when they are ready. In the mean time, there is a branch being worked within the gnuada project SVN structure on sourceforge.

If you have a large library you want to bind to, contributing to SWIG may be an overall cost neutral approach and be helpful in the long term.

Having said that, many C libraries have gotten so cluttered with defines and decl specs and exports of various flavors that I suspect that some libraries will forever resist effective automated binding generation.

4) Something else similar. Specifically, GtkAda has a perl script that is semi-specific to Gtk that helps in the creation of bindings. It is sort of a middle ground and is an effective approach in some special cases.

Interfacing to C macros

From: Maciej Sobczak <maciej@msobczak.com>
Newsgroups: comp.lang.ada
Subject: Interfacing to C: API with macros
Date: Tue, 12 Jun 2007 13:17:29 −0700

Yet Another Problem (YAP): consider a C function that is defined together with some helper macros. Motivating example: select(2) system call, with its FD_XXX helper macros.

(it is unspecified whether FD_XXX helpers are macros or functions, but we can assume the worst)

There is no way to pragma Import(C, FD_SET) and the binary layout of fd_set data structure is not specified, so we cannot fake it with Ada.

Again, the straightforward solution: thin wrapper in C, that itself is simple enough to be easily imported by Ada code. […]

From: Lutz Donnerhacke <lutz@iks-jena.de>
Newsgroups: comp.lang.ada
Subject: Re: Interfacing to C: API with macros
Date: Tue, 12 Jun 2007 20:25:17 +0000 UTC
Organization: IKS GmbH Jena

Have a look at the semantics of those macros, you will notice, that they are a application level language in C hiding the expressiveness holes of the language.

Now you have a higher level language and you are expected to use it!

We know that fdset_t is a bitarray on all known systems. Therefore

```
type fdset is Array (0..1023) of Boolean;
pragma Pack(fdset);
```

and you are done.

If you try to meet your counterpart at the most evil place, your program will look as ugly as the counterpart. Use the abstraction of your language! Program what they mean, not what they code.

From: Tom Moran <tmoran@acm.org>
Newsgroups: comp.lang.ada
Subject: Interfacing to C: API with macros
Date: Tue, 12 Jun 2007 19:18:55 −0500

If you are interfacing to something that isn't documented, then you will indeed have to experimentally determine the interface, or make a wrapper from unknown to known interface spec.

Usually the interface is documented (though too often erroneously), though you may have to look carefully to see, for instance, what some bit-field packing macro is doing. Once you know, you can write a direct Ada interface. Claw, for instance, uses a great number of Windows API calls, and it uses no wrappers in C (though the whole point of Claw is to give an Ada-flavor wrapper to the Windows API). Many APIs in the Windows world also have typelib descriptions that can be used to automatically generate an Ada wrapper.

From: Steve <steved94@comcast.net>
Newsgroups: comp.lang.ada
Subject: Re: Interfacing to C: API with macros
Date: Wed, 13 Jun 2007 07:06:15 −0700

> Looks like in order to write any nontrivial system software the poor Ada programmer has to start with an awful lots of wrappers — where is maintainability and productivity when you need them? ;-)

Fortunately the Ada programmer gains enough efficiency in other areas to offset the time generating wrappers. Some tools (ie: c2ada) are available for automatically generating wrappers, but do not in general do 100% of the job.

System IO and text files

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: Mixing reading and writing to a text file
Date: Thu, 2 Aug 2007 20:26:33 −0500

> I'm facing a situation, where I need to access a text file for both input and output.

The only way to read and write the same file is using Ada.Streams.Stream_IO. Use Reset or Set_Mode to change from Reading to Writing and vice versa.

> This file is used for storing a sequence of different records (of various lengths — it's not possible to define the upper bound) defining operations that need to be performed. Some errors could occur and the processing might be aborted. In this case it is crucial to store state of the processing (in the simplest case only the number of the last record processed) must be saved, so that it may be resumed later.

> The task seems simple as I have complete control over the file format. I'd rather use human-readable format but a binary one is also acceptable.

> The problem I have is with writing the state information back to file. Ada.Text_IO and Ada.Streams.Stream_IO only allow opening file for input or for output. In the latter case the file is being truncated (as far as I understand the ARM).

The ARM was screwed up in Ada 95 vis-a-vis Ada.Streams.Stream_IO. This was fixed in the Amendment (and the fix is supposed to apply to Ada 95 compilers as well). Specifically, stream files are *not* truncated when they are opened for output (otherwise it would be virtually impossible to use the positioning functions to write a stream file).

But a warning: almost all compilers got this wrong when we tested them while we were working on the Amendment. (The main reason that we were willing to change it was that virtually every compiler tested did something different.) So it is not impossible that your implementation gets this wrong in some way. But if it does, that is a compiler bug, not a language issue. Report it to your implementer.

Also note that you don't need to Open the file to change the mode of a stream file; Set_Mode should do the job. And even the buggy Ada 95 manual didn't imply that Set_Mode should truncate, so that ought to work (but again, not all compilers get this right — unfortunately, there was no ACATS test for it).

From: Marcin Simonides <msimonides@power.com.pl>
Newsgroups: comp.lang.ada
Subject: Re: Mixing reading and writing to a text file
Date: Mon, 06 Aug 2007 00:35:54 −0700

Thanks for clarification. I wrote a simple test that opens an In_File and then Resets it as Out_File and writing works as expected — data is overwritten over the bytes that I wish to change and there is no truncation (the compiler is GNAT GPL 2007).

(I have read A.8.2 File Management and only skimmed over description of Ada.Streams.Stream_IO, so this has been mostly an RTFM issue :)).

Self pointer in limited record

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Self pointer in limited record
Newsgroups: comp.lang.ada
Date: Thu, 5 Jul 2007 10:22:59 +0200

> Consider:

```
> type T_Access is access all T;
> type T is  new
    Ada.Finalization.Limited_Controlled
    with record
>    Self : T_Access :=
    T'Unchecked_Access;
>    -- more components
>    -- …
> end record;
```

> I have seen this pattern repeatedly. What is the use for Self?

The pattern (also called Rosen trick) is used:

1. to have mutable arguments of functions:

```
function Search (X : T; K : Key) return
         Value is
  Object : T renames X.Self.all;
begin
  … -- Modify the search cache
         associated with X via
  -- mutable Object view
```

2. to re-dispatch from primitive operations:

```
type T_Access is access all T'Class;  --
Note 'Class
type T is  new
Ada.Finalization.Limited_Controlled with
record
Self : T_Access :=
T'Unchecked_Access;
procedure Bar (X : T); -- Primitive
procedure Foo (X : T); -- Primitive
procedure Foo (X : T) is
Object : T'Class renames X.Self.all;
begin
  Bar (X); -- This does not dispatch!
  Bar (Object); -- This dispatches
  Bar (T'Class (X));  -- This dispatches
as well
```

The later (view conversion) should better be removed from the language, so I always prefer Rosen trick for such purpose.

Both defeat the type system in some sense and potentially indicate a design problem.

> And why is it Unchecked?

Because of accessibility rules.

From: Adam Beneschan <adam@irvine.com>
Newsgroups: comp.lang.ada
Subject: Re: Self pointer in limited record
Date: Fri, 31 Aug 2007 10:37:10 −0700

> Then you have seen illegal Ada code repeatedly. I wish this were possible myself. Or more simply:

> type T is -- limited or not

> Self : access T := T'Unchecked_Access;

> …

> end;

> But the compiler will remind you that Unchecked_Access is not available for types.

No, this can be legal. Normally, you can't apply 'Unchecked_Access to a type name. But within the definition of a type T, the use of T in a context like this refers to the "current instance"; that is, it will refer to whatever object is declared with that type. So if you later declare "X : T;", then the T in T'Unchecked_Access will be replaced by X for that declaration (and X.Self will thus point to X). See 8.6(17). However, it's only legal if T is limited (in Ada 2005, the rule is slightly more restrictive), because 'Unchecked_Access can only be applied to an aliased entity, and 3.10(9) says that the current instance of a *limited* type is defined to be aliased.

Date: Sat, 01 Sep 2007 15:33:09 +0200
From: Georg Bauhaus <bauhaus.rm.tsoh@maps.futureapps.de>
Newsgroups: comp.lang.ada
Subject: Re: Self pointer in limited record

> T cannot be non-limited, because otherwise passing it by copy would make rubbish out of Self. In any case it would make little sense if not access T'Class.

I think there is an interesting use of a .Self pointer of simple limited records with state variables of packages.

Say a procedure in a package P controls the state of some variable in the package's body. The state variable is of type access T, where T is a limited record type declared in P. The purpose of the state variable is to remember the object of type T that will be the target of subsequent package operations.

A natural way to remember a particular object is to point to the object. But package clients should not have to worry about this pointing mechanism. So the package shouldn't declare a public access type used for internal mechanism only. Instead there is a procedure of one T argument that can be called by clients when they want to indicate the object to be used in subsequent P operations.

This is where .Self can be used. We cannot take 'Access or 'Unchecked_Access of subprogram arguments unless they are aliased (such as those of type T'Class). But the .Self component of the subprogram's argument supplies the access value that is needed for storing a pointer to the argument in the package body.

```
package P is
  type T is limited private;
  procedure Choose (Selected: in out T);
  -- sets target T for subsequent
     operations
  procedure Work;
  private
    type Bits is array(0 .. 15) of Boolean;
    type T_Access is access all T;
    type T is limited record
      Self: T_Access :=
              T'Unchecked_access;
      Slots: Bits;
    end record;
end P;
package body P is
  Current: T_Access;  - - state variable,
              target object
  procedure Choose (
              Selected: in out T) is
  begin
    Current := Selected.Self;  -- here
  end Choose;
  procedure Work is
  begin
    Current.Slots(3) := not
              Current.Slots(3);
  end Work;
end P;
```

[…]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Self pointer in limited record
Newsgroups: comp.lang.ada
Date: Sat, 1 Sep 2007 15:46:27 +0200

> (The true O-O programmer might suggest that we should simply pass an additional object-as-module parameter to every package subprogram…)

Yes, it is better to keep packages stateless.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: Re: Self pointer in limited record
Newsgroups: comp.lang.ada
Date: Sat, 1 Sep 2007 18:03:00 +0200

> Though by using tagged objects for a module and not a stateful package, you will dismiss a few properties of packages that can be helpful when modeling singleton modules:

> 1.- If there is just one object in the problem domain a package is a perfect match and will be straight forward, safe, and simple to implement. No need to worry about static and dynamic scopes of module-objects passed around because there is just this one named package for the problem domain object.

Yes, yes, but this is a different case. Dealing with singletons, I probably would use a [stateful] package instead of objects. Types presume multiple instances of. Singleton in OO breaks this concept. Ada offers a cleaner alternative. Why should we force it into a type? Let it be a package.

I think the empiric rule could be: if a package is used to declare types, then it should have no mutable state. Otherwise it should not have type declarations.

> 2.- Nesting packages is an option, a distinguishing feature of Ada IMO; a package nested inside a subprogram is a simple solution to the life cycle problem of module style objects.

Hmm, a subprogram has all properties of a package. So there is no obvious reason why nested package (except instances of generic packages, of course), might be useful there.

BTW, I guess child and separate packages might probably replace nested packages. Excluding generics, I mean. Maybe if there were no generic packages we could drop them altogether.

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 and events for the international Ada community* at: http://www.cs.kuleuven.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programs, URLs, etc. and are updated regularly.

2007

October 02-05 — 23rd IEEE **International Conference on Software Maintenance** (ICSM'2007), Paris, France. Topics include: software and systems maintenance, evolution, and management.

October 03-05 — 10th **Italian Conference on Theoretical Computer Science** (ICTCS'2007), Rome, Italy. Topics include: formal languages; formal methods and model checking; models of concurrent and distributed systems; principles of programming languages; program analysis and transformation; specification, refinement and verification; etc.

October 11 — Automotive SPIN Italy - 2nd **Workshop on Automotive Software**, Milan, Italy. Topics include: any aspects of Software Process and Software Engineering in the Automotive Domain, such as Tool and technical solutions supporting software process improvement, Software Certification issues in automotive, Safety implication for automotive software, etc.

October 11-12 — 7th **International Conference on Quality Software** (QSIC'2007), Portland, Oregon, USA. Topics include: Software quality (review, inspection and walkthrough, reliability, safety and security, ...); Evaluation of software products and components (static and dynamic analysis, validation and verification); Formal methods (program analysis, model checking, ...); Applications (component-based systems, distributed systems, embedded systems, safety critical systems, ...); etc.

October 11-12 — 12th **Nordic Workshop on Secure IT Systems** (NordSec'2007), Reykjavik, Iceland. Topics include: Language-based Techniques for Security, Security Education and Training, Trust and Trust Management, etc.

October 15-17 — 2007 **International Multiconference on Computer Science and Information Technology** (IMCSIT'2007), Wisla, Poland.

- ☺ Oct 15-17 — 1st **Workshop on Advances in Programming Languages** (WAPL'2007). Topics include: Compiling techniques; Domain-specific languages; Formal semantics and syntax; Generative and generic programming; Languages and tools for trustworthy computing; Language concepts, design and implementation; Metamodeling and modeling languages; Model-driven engineering languages and systems; Practical experiences with programming languages; Program analysis, optimization and verification; Program generation and transformation; Programming tools and environments; Proof theory for programs; Specification languages; Type systems; etc
- ☺ October 16 — **International Workshop on Real-Time Software** (RTS'2007). Topics include: real-time system development, real-time scheduling, safety, reliability, dependability, fault-tolerance, standards and certification, software development tools, model-based development, automatic code generation, real-time systems curricula, etc.

October 15-19 — 21st **Brazilian Symposium on Software Engineering** (SBES'2007), Joao Pessoa/PB, Brazil. Topics include: Component-based software engineering; Empirical software engineering and metrics; Generative or transformation-based software development; Model driven development; Software architecture, design and frameworks; Software engineering for embedded and real-time Software; Software engineering tools and environments; Software safety, dependability, and reliability; Software maintenance and reverse engineering; Software analysis and design methods; Software engineering metrics; Software quality; Software reuse; Software testing and analysis; Software verification, validation and inspection; etc.

☺ October 18 4th **Workshop on Programming Languages and Operating Systems** (PLOS'2007), Stevenson, WA, USA. Topics include: critical evaluations of new programming language ideas in support of OS construction; type-safe languages for operating systems; language-based approaches to crosscutting system concerns, such as security and run-time performance; language support for system verification; etc.

☺ October 21-25 22nd **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2007), Montreal, Canada. Topics include: programmer productivity, secure and reliable software, changing hardware platforms, ultra-large scale systems, improve programming languages, refine the practice of software development, etc.

- ☺ October 21 6th **"Killer Examples" workshop**. Theme: "Process in OO Pedagogy". Topics include: "killer" examples of teaching the process of programming; "killer" examples of teaching OO modeling and programming; analyses of the process which students use to solve problems; what are needs of industry - what will keep students competitive in the workplace?; etc.
- October 21 3rd **Workshop on Library Centric Software Design** (LCSD'2007). Topics include: Design and implementation of libraries; Program and system design based on libraries; Evolution, refactoring, and maintenance of libraries; Design of language facilities and tools in support of library definition and use; Validation, debugging, and testing of libraries; Extensibility, parameterization, and customization; Specification of libraries and their semantics; Assessing quality of libraries; Using several libraries in combination; etc.
- ☺ October 22 1st **Workshop on Programming Languages and Integrated Development Environments** (PLIDE'2007). Topics include: techniques for supporting languages in IDEs.

☺ October 22 2nd **International Conference on System Safety** 2007, London, UK. Includes: talk on "Certification of Object Oriented Programs", by Robert Dewar, AdaCore.

October 26 1st **York Doctoral Symposium on Computing** (YDS'2007), York, UK. Topics include: High integrity system engineering, within the context of Formal methods, Verification and Formal verification, Theorem proving, Model checking, Testing; Information systems, within the context of Formal Methods in Software Engineering, Model Driven Development, Object-Oriented Modelling and Development, Systems Engineering Methodologies, Modelling Formalisms (Languages and Notations), CASE Tools for System Development, Security, Component-Based Development, Software Architecture, Software Engineering for Concurrent and Distributed Systems, Software Quality, Software Verification (Validation and Inspection), ...; IT Security; Programming languages and systems; Real-time systems; etc.

October 27-31 14th **Working Conference on Reverse Engineering** (WCRE'2007), Vancouver, British Columbia, Canada. Theme: "Using evolution history for reverse engineering". Topics include: Mining software repositories; Empirical studies in reverse engineering; Program comprehension; Redocumenting legacy systems; Reverse engineering tool support; Software architecture recovery; Program analysis and slicing; Program transformation and refactoring; etc.

☺ October 30-31 4th **Workshop on Object-oriented Modeling of Embedded Real-Time Systems** (OMER-4), Paderborn, Germany. Topics include: Architectures/frameworks for platform independent, reusable software components; Formal verification at the model and code level; Software components as products; Software quality; Standards and guidelines (e.g., AUTOSAR, IEC 61508, MISRA, UML, ...); Respective trends in automotive software development; etc.

October 30-31 IEEE **International Conference on Software - Science, Technology & Engineering** (SwSTE'2007), Herzliya, Israel. Topics include: Verification, validation, and testing; Software engineering education and training; Safety, reliability, and fault tolerance; Embedded systems and real-time software; Open-source software; Analysis, design, and implementation; Modeling languages and tools; Programming languages and environments; Analysis and design patterns; Maintenance, reuse, and evolution; etc.

♦ Nov 04-08 2007 ACM **SIGAda Annual International Conference** (SIGAda'2007), Fairfax, Virginia, USA (a suburb of Washington, DC). Sponsored by ACM SIGAda, in cooperation with SIGAPP, SIGCAS, SIGCSE, SIGPLAN, SIGSOFT, Ada-Europe, and Ada

Resource Association. Topics include: Safety, security and high integrity development issues; Language selection for a high reliability system; Use of ASIS for new Ada tool development; Mixed-language development; High reliability software engineering education; High reliability development experience reports; Static and dynamic code analysis; Use of new Ada 2005 features/capabilities; etc. Deadline for submissions: October 15, 2007 (nominations for SIGAda Awards).

☺ Nov 08-09 NIST **SAMATE Static Analysis Summit II**. Topics include: basic research, applications, experience, or proposals relevant to static analysis tools, techniques, and their evaluation.

November 05-09 18th IEEE **International Symposium on Software Reliability Engineering** (ISSRE'2007), Trollhaettan, Sweden. Topics include: Reliability, availability and safety of software systems; Quality/reliability-related security issues; Verification and validation; Industrial best practices; Empirical studies of those topics; etc. Includes workshop on Automotive Software Reliability. Deadline for early registration: October 7, 2007

November 07-09 6th **International Conference on Software Methodologies, Tools, and Techniques** (SoMeT'2007), Rome, Italy. Topics include: Software methodologies, and tools for robust, reliable, non-fragile software design; Automatic software generation versus reuse, and legacy systems, source code analysis and manipulation; Intelligent software systems design, and software evolution techniques; Software optimization and formal methods for software design; Software security tools and techniques, and related Software Engineering models; End-user programming environment; Software Engineering models, and formal techniques for software representation, software testing and validation; etc.

November 14-15 9th **International Conference on Formal Engineering Methods** (ICFEM'2007), Boca Raton, Florida, USA. Topics include: Abstraction and refinement; Tool development and integration for system design and verification; Techniques for specification, verification and validation; Techniques and case studies for correctness by construction; Applications in real-time, hybrid and critical systems; Development methodologies with their formal foundations; etc. Deadline for early registration: October 14, 2007.

November 14-16 10th IEEE **International Symposium on High Assurance Systems Engineering** (HASE'2007), Dallas, Texas. Topics include: Design and development of highly reliable, survivable, secure, safe, and time-assured systems; Policies for reliability, safety, security, integrity, privacy, and confidentiality of high assurance systems; Formal specification, specification validation, testing, and model checking for high assurance systems; High assurance software architectures and design; Case studies, experiments and tools for high assurance systems; etc.

November 22-23 7th **International Workshop on Advanced Parallel Processing Technologies** (APPT'2007), Guangzhou, China. Topics include: Parallel/distributed system architectures; Middleware, software tools and environments; Parallelizing compilers; Software engineering issues; Task scheduling and load balancing; Security in networks and distributed systems; Fault tolerance and dependability; etc.

☺ November 25-30 9th **International Symposium on Distributed Objects and Applications** (DOA'2007), Vilamoura, Algarve, Portugal. Topics include: Application case studies of distribution technologies; Interoperability with other technologies; Reliability, fault tolerance, quality-of-service, and real time support; Scalability and adaptivity of distributed architectures; etc.

November 26-27 4th **International Workshop on Rapid Integration of Software Engineering techniques** (RISE'2007), Luxembourg, Luxembourg. Topics include: Software reuse, Lightweight or practice-oriented formal methods, Software processes and software metrics, Software patterns, Design by contract, Defensive programming, Software entropy and software re-factoring, Programming languages, Software dependability and trustworthiness, High-availability or mission-critical systems, Resilient business and grid applications, Embedded systems and applications, Development environments, etc.

Nov 29 – Dec 01 5th **Asian Symposium on Programming Languages and Systems** (APLAS'2007), Singapore. Topics include: foundational and practical issues in programming languages and systems, such as semantics, type systems, language design, program analysis, optimization, software security, safety, verification, compiler systems, programming tools and environments, etc.

☺ December 03-06 28th IEEE **Real-Time Systems Symposium** (RTSS'2007), Tucson, Arizona, USA. Topics include: all aspects of real-time systems design, analysis, implementation, evaluation, and case-studies.

☺ December 03-06 8th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2007), Adelaide, Australia. Topics include: Formal methods and programming languages, Software tools and environments, Component-based and OO Technology, Parallel/distributed algorithms, Task mapping and job scheduling, High-performance scientific computing, etc.

December 03-12 3rd **International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering** (CISSE'2007), Internet. Includes 4 e-conferences, among others the International Conference on Systems, Computing Sciences and Software Engineering (SCSS'2007) with topics: Programming Models and tools, Parallel and Distributed processing, Modeling and Simulation, Embedded Systems and Applications, Programming Languages, Object Based Software Engineering, Parallel and Distributed Computing, Real Time Systems, Multiprocessing, etc. Deadline for submissions: October 5, 2007.

☺ December 06 **Journée Ada-France**, Brest, France. Theme: "Méthodes, processus, modèles et outils pour l'ingénierie du logiciel embarqué temps réel critique". Topics include: des expérimentations d'outils, de modèles et/ou de méthodes utilisés ou susceptibles d'être utilisés pour la réalisation de systèmes embarqués temps réel critiques. Deadline for submissions: October 20, 2007 (presentation proposals).

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

December 17 BCS-**FACS 2007 Christmas Workshop: Formal Methods in Industry**, London, UK. Topics include: industrial uses of formal methods, lessons learned from applying formal methods in industry, industrial case studies demonstrating the use of formal methods, use of formal methods tools in industry, opportunities for applying formal methods in industry, etc. Deadline for submissions: October 1, 2007

December 18-21 14th IEEE **International Conference on High Performance Computing** (HiPC'2007), Goa, India. Topics include: Parallel and Distributed Algorithms, Parallel Languages and Programming Environments, Scheduling, Scientific/Engineering Applications, Software Support, etc.

2008

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

January 13 2008 **International Workshop on Foundations of Object-Oriented Languages** (FOOL'2008), San Francisco, California, USA. Topics include: language semantics, type systems, program analysis and verification, concurrent and distributed languages, language-based security issues, etc. Deadline for submissions: October 8, 2007.

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

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

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

☺ March 04-07 CISIS2008 - **International Workshop on Multi-Core Computing Systems** (MuCoCoS'2008), Barcelona, Spain. Topics include: programming languages and models; performance modeling and evaluation of multi-core systems; tool-support for multi-core systems; compilers, runtime and operating systems; etc. Deadline for paper submissions: October 10, 2007. Deadline for early registration: December 15, 2007

☺ March 12-14 SIAM **Conference on Parallel Processing for Scientific Computing** (PP'2008), Atlanta, Georgia, USA. Topics include: Programming languages, models, and compilation techniques; The transition to ubiquitous multicore/manycore processors; Tools for software development and performance evaluation; Parallel computing in industry; Distributed/grid computing; Fault tolerance; etc. Deadline for submissions: October 1, 2007 (minisymposium proposals), October 8, 2007 (abstracts)

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

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

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

Mar 29 – Apr 06 **European Joint Conferences on Theory and Practice of Software** (ETAPS'2008), Budapest, Hungary. Deadline for submissions: October 5, 2007 (research and tool paper abstracts), October 12, 2007 (research and tool papers).

Mar 31 – Apr 04 7th **International Conference on Aspect-Oriented Software Development** (AOSD'2008), Brussels, Belgium. Deadline for submissions: October 5, 2007 (research paper abstracts), October 12, 2007 (research papers), October 19, 2007 (workshops), November 21, 2007 (tutorials, demos, industry track submissions)

☺ April 01-04 3rd **European Conference on Computer Systems** (EuroSys'2008), Glasgow, UK. Topics include: All areas of operating systems and distributed systems; Systems aspects of: Dependable computing, Parallel and concurrent computing, Distributed algorithms, Programming language support, Real-time and embedded computing, Security, ...; Experience with existing systems; Reproduction or refutation of previous results; Negative results; Early ideas. Deadline for submissions: November 23, 2007 (workshops).

April 01-04 12th **European Conference on Software Maintenance and Reengineering** (CSMR'2008), Athens, Greece. Theme: "Developing Evolvable Systems". Topics include: Software migration strategies and technologies; Empirical studies in maintenance and reengineering; Experience reports on evolution, maintenance and reengineering; Education in maintenance and reengineering; etc. Deadline for submissions: October 12, 2007 (papers), October 19, 2007 (doctoral symposium papers, industrial track papers, workshops, tool sessions)

☺ April 14-18 22nd IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2008), Miami, Florida, USA. Topics include: all areas of parallel and distributed processing, such as Applications of parallel and distributed computing; Parallel and distributed software, including parallel programming languages and compilers, runtime systems, middleware, libraries, and programming environments and tools, etc. Deadline for submissions: November 15, 2007 (tutorials)

May 07-09 7th **European Dependable Computing Conference** (EDCC-7), Kaunas, Lithuania. Topics include: Architectures for dependable systems; Fault tolerant distributed systems; Fault tolerance in real-time

systems; Hardware and software testing, verification, and validation; Formal methods for dependability; Safety-critical systems; Software reliability engineering; Software engineering for dependability; etc.

☺ May 10-18 30th **International Conference on Software Engineering** (ICSE'2008), Leipzig, Germany. Topics include: Software components and reuse, Theory and formal methods, Engineering secure software, Software dependability, safety and reliability, Reverse engineering and maintenance, Software economics and metrics, Empirical software engineering, Engineering of distributed/parallel software systems, Engineering of embedded and real-time software, Software tools and development environments, Programming languages, etc. Deadline for submissions: October 12, 2007 (Education Papers, Tutorial Proposals, Workshop Proposals), November 30, 2007 (Research Demonstrations), December 14, 2007 (Doctoral Symposium)

May 26-30 15th **International Symposium on Formal Methods** (FM'2008), Turku, Finland. Topics include: all aspects of formal methods research, both theoretical and practical, in particular the experience of applying formal methods in practice.

June 04-06 10th IFIP **International Conference on Formal Methods for Open Object-based Distributed Systems** (FMOODS'2008), Oslo, Norway. Topics include: Semantics and implementation of object-oriented programming and (visual) modelling languages; Formal techniques for specification, design, analysis, verification, validation and testing; Applications of formal methods; Experience report on best practices and tools; etc. Deadline for submissions: January 8, 2008 (abstracts), January 15, 2008 (papers)

♦ June 16-20 13th **International Conference on Reliable Software Technologies** – Ada-Europe 2008, Venice, Italy. Organized and sponsored by Ada-Europe, in cooperation with ACM SIGAda (approval pending). Deadline for submissions: November 4, 2007 (papers, tutorials, workshops), January 13, 2008 (industrial presentations)

June 17-20 28th **International Conference on Distributed Computing Systems** (ICDCS'2008), Beijing, China. Topics include: theoretical foundations, reliability and dependability, security, middleware, etc. Deadline for submissions: November 15, 2007 (papers).

June 30 – July 02 13th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2008), Madrid, Spain.

☺ June 30 – July 04 **Technology of Object-Oriented Languages and Systems** (TOOLS Europe'2008), Zurich, Switzerland. Topics include: all modern approaches to software development, with a special but not exclusive emphasis on O-O and components.

July 06-13 35th **International Colloquium on Automata, Languages and Programming** (ICALP'2008), Reykjavik, Iceland. Topics include: Principles of Programming Languages; Formal Methods and Model Checking; Models of Concurrent and Distributed Systems; Models of Reactive Systems; Program Analysis and Transformation; Specification, Refinement and Verification; Type Systems and Theory; Foundations of Secure Systems and Architectures; Specifications, Verifications and Secure Programming; etc. Deadline for submissions: October 31, 2007 (workshops), February 10, 2008 (papers).

☺ July 07-11 22nd **European Conference on Object Oriented Programming** (ECOOP'2008), Paphos, Cyprus. Topics include: analysis, design methods and design patterns; concurrent, real-time or parallel systems; distributed systems; language design and implementation; programming environments and tools; type systems, formal methods; compatibility, software evolution; components, modularity; etc. Deadline for submissions: December 19, 2007 (papers).

July 07-13 20th **International Conference on Computer Aided Verification** (CAV'2007), Princeton, USA. Topics include: Algorithms and tools for verifying models and implementations, Program analysis and software verification, Applications and case studies, Verification in industrial practice, etc. Deadline for submissions: October 15, 2007 (workshops), January 28, 2008 (papers, CAV Award nominations).

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

Towards Certification of Object-Oriented Code with the GNAT Compiler

Javier Miranda

Instituto Universitario de Microelectrónica Aplicada. Universidad de Las Palmas de Gran Canaria, Canary Islands, Spain; email: jmiranda@iuma.ulpgc.es

Courant Institute: Computer Science Department. New York University. 251 Mercer Street, NY 10012.

AdaCore. 104 Fifh Avenue, 15th floor. New York, NY 10011.

Abstract

Dynamic binding, the ability to link at runtime a method call with a subprogram that depends on the class of the object, is strongly discouraged by current standards for avionics airborne systems. This is partly due to dynamic dispatching, the technique commonly used by most OO compilers to implement dynamic binding. In this paper we present some enhancements to the GNAT technology that will help the avionic industry take advantage of the full benefits of the OO techniques with Ada without the inconveniences associated with dynamic dispatching.

Keywords: Dynamic dispatching, Airborne Systems, High-Integrity, Ada 2005, Tagged Types, Abstract Interface Types, GNAT.

1 Introduction

Reliable software construction has evolved considerably in the last two decades. There is currently a trend towards the use of Object Oriented Techniques (OOT) in the construction of High-Integrity Software Systems, such as avionics airborne systems. In this domain, one of the objectives of the forthcoming revision of the DO-178B standard [9] is to address the use of OOT and their associated development processes in the avionics industry. A preliminary document of the future DO-178C [3] provides a comprehensive analysis on the safety concerns associated with OO techniques in the context of DO-178B.

Since the emergence of the DO-178B standard, Ada [11] has been one of the few languages of choice for the construction of airborne systems, thanks to its clear semantic definition and strong typing model. It has been used successfully in many major aeronautics projects (Boeing 777, A340, and more recently Boeing 787, A380 and A400M). In recent years Ada has evolved to fulfill the requirements of modern software industry incorporating object-oriented features into its original type model. The Ada 95 standard added to Ada tagged types, single inheritance, polymorphism, and dynamic dispatching. The latest revision of the language, known as Ada 2005, adds multiple inheritance of abstract interface types and numerous other object-oriented programming idioms.

A crucial element of Object Oriented Programming (OOP) is *dynamic binding*, that is the ability to link at runtime a method call with a subprogram based on the class of the object on which the method is invoked. In their current form, DO-178B is wary of dynamic binding: its use is not formally banned, but it is strongly discouraged by DO-248B [10, FAQ 34]. This is partly due to *dynamic dispatching*, the technique used to implement dynamic binding in most compiled OO languages. Although solutions to these issues are emerging, they are not yet fully established.

In this paper we present several ongoing research projects whose main purpose is to facilitate the certification of OO code written in Ada with the GNAT compiler. In Section 2 we summarize inheritance and polymorphism concepts and their common implementation by means of dispatching tables. In Section 3 we describe the main problems of dynamic dispatching in the context of safety and security systems. In Section 4 we present four enhancement projects of the GNAT technology that will help to certify OO Ada code for High-Integrity systems. We close with some conclusions and the bibliography.

2 Inheritance and Polymorphism in Ada

Inheritance was originally viewed as a mechanism for sharing code and data definitions. Multiple inheritance was viewed as a mechanism for constructing a subclass implementation from multiple superclass implementations. As understanding of OO modeling has matured, however, the focus has increasingly been on the specification of interfaces and the specification of interfaces as contracts between clients and implementers. Multiple inheritance is currently used primarily as a means of classifying entities that logically belong to more than a single category. As a result, languages such as Java [5] and Ada 2005 [11] only support multiple inheritance of interfaces and rely on delegation to achieve the effects of multiple implementation inheritance.

In the context of High-Integrity Systems, the general OO avionics guidance [3, Section 3.4] makes a strong distinction between multiple inheritance of specifications and multiple inheritance of implementations as provided by C++, and recommends use of multiple implementation inheritance only for level D software. (The DO-178B

standard defines five levels of safety-criticality, ranging from Level A at the most critical, to Level E at the least critical; the top three safety levels are of particular interest to Ada developers.)

Polymorphism permits instances of a subclass to be assigned to variables of a superclass, and it is used to specify generic algorithms that are common to a given hierarchy of classes. In this context, dynamic binding ensures that the method executed by a call to a polymorphic object is that associated with the object's run time type. Conceptually, at run-time there is a single dispatch routine containing a pair of nested case statements [3, Section 3.3.2]:

```
case <Object'Run-Time-Type> is
   ...
   when <Class-N> =>
     case <Method'Signature> is
       ...
       when <Method-N> =>
         call <Method-N> defined by <Class-N>
      end case
end case
```

In practice, dynamic binding is typically implemented using *Dispatch Tables*, which introduce a small and fixed overhead (cf. Figure 1). Each object instance has a hidden component (the *Vtable* pointer in C++ and Java, or the *Tag* in Ada) that references a dispatch table with the run-time type's method signatures. (For efficiency reasons the method signature is generally the address of the target method.) At the point of a dispatching call the compiler generates code that uses this hidden component to (1) get the dispatch table associated with the object, (2) index it by a number associated with the method signature (a constant known at compile time), and (3) make an indirect invocation of the target method.

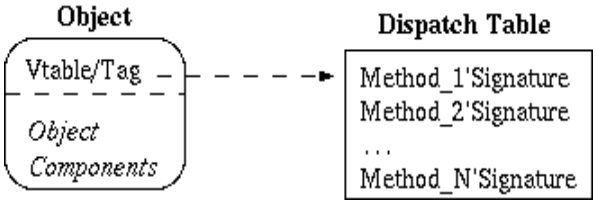

Figure 1 Object Layout

In Ada, subprograms declared together with a tagged type in the same package and having at least one parameter (or result) of the tagged type are called primitive operations of the type. A call to such an operation is not necessarily dispatching however. The call will only dispatch when invoked with an actual parameter whose type is the class-wide type of the associated type class (*T'Class* denotes the entire set of types in the class of *T*). This flexibility can be used to limit the number of dispatching calls, thereby limiting their associated certification cost. Prevention of dispatching can be also enforced by the use of pragma Restrictions (No_Dispatch).

This flexibility is not available in Java where all operation invocations are dispatching (unless a routine is declared as final, which allows the compiler to perform various optimizations knowing the primitive cannot be overridden). It is available in C++, but at the cost of forcing the programmer to indicate whether an operation itself (not a specific call) is virtual. A virtual operation will potentially always dispatch while a non-virtual one will never dispatch. C++ compilers are allowed to optimize dispatching calls into regular calls when the context permits, but this is not under the control of the developer.

3 Problems with Dynamic Dispatching in High-Integrity Systems

Dynamic dispatching has several safety and security problems, namely:

- **Initialization**: how can we prove that dispatch tables and *Tag* fields are initialized correctly?
- **Modification**: how can we prove that dispatch table and *Tag* values are not updated maliciously or unintentionally during the execution of a program?
- **Tools**: being dynamic dispatching invisible at the source level, how can we use source-based tools in the presence of dynamic dispatching for code coverage?

Demonstrating correct dispatch-table initialization at object-level is akin to the problem of showing that the linker produces a correct executable from the object files it links. This problem is part of the control coupling objective in DO-178B parlance and is addressed by either verifying the correctness of the final result by hand of by employing a qualified tool that performs such verification [12].

If one can ROM dispatch tables or place them in OS-guarded read-only memory the need to verify that dispatch tables are unchanged during a program's execution disappears. Unfortunately, an object's *Tag* field cannot typically be placed into read-only memory and the costs of demonstrating at object-code level that these fields are unchanged during a program's execution remain. Such modifications could occur because of a rogue pointer or buffer overflow in assembly or C/C++ code that may be part of the application or by other accidental or malicious means.

In the following section we present several enhancements that will help to workaround these problems with the GNAT technology.

4 Towards certification of dispatching calls with the GNAT compiler

In order to certify dispatching calls in High-Integrity Software the following concerns of the general OO avionics guidance must be answered by the compiler [3]:

- **Stack Analysis**: ``Stack overflow errors are listed in section 6.4.3f of DO-178B as errors that are typically found in requirements-based hardware-software integration testing. Timing and stack

analysis are complicated by certain implementations of dynamic dispatch. If polymorphism and dynamic binding are implemented, stack size can grow, making analysis of the optimal stack size difficult" [3, Section 2.3.3.1.3].

- **Object Code Traceability**: ``Everywhere concerns about source code to object code traceability and timing analysis dictate, the compiler vendor may be asked to provide evidence of deterministic, bounded mapping of the dispatched call. If the evidence is not available from the compiler vendor, it may be necessary to examine the structure of the compiler-generated code and data structures (e.g., method tables) at the point of call" [3, Section 3.3.4.3].
- **Structural Coverage**: ``Many current Structural Coverage Analysis tools do not ``understand" dynamic dispatch, i.e. do not treat it as equivalent to a call to a dispatch routine containing a case statement that selects between alternative methods based on the run-time type of the object. (IL 55)" [3, Section 2.3.3.1.2].

In this section we present several ongoing enhancements to the GNAT technology that will help to solve these problems. The concern of stack-analysis has been already addressed by GNAT with the gnatstack tool (work described in a separate paper [1]). In the following sections we present three additional enhancement projects: in Section 4.1 we present the visualization of the dispatch tables at the source level; in Section 4.2 we present the static allocation of dispatch tables. These projects are currently at their final stage. Finally in Section 4.3 we present a new project that expands dispatching calls into case statements.

4.1 Dispatch Table Visualization

The first enhancement of the GNAT technology addresses the correct initialization of the dispatch table. The compiler has being improved to leave the initialization of the dispatch tables visible at source level and hence to support the DO-178B traceability requirement. Using a switch the compiler currently generates Ada-like code that allows to see the expansion performed by the frontend. As part of this project, the output associated with the construction of dispatch tables has been improved to facilitate the use of source-based tools based on static control flow to verify their correct initialization. Such Ada-like code can be also visualized during debugging using another compiler switch. Let us consider the following Ada 2005 example to present this new output:

```
package Iface is
   type Writable is interface;
   procedure Read
      (Obj : Writable; Data : out Integer) is abstract;
   procedure Write
      (Obj : Writable; Data : in Integer)  is abstract;
end Iface;
```

Package Iface contains the declaration of the abstract interface type Writable that has two abstract primitives: Read and Write.

```
with Iface; use Iface;
package Pkg is
   type Root is tagged ... ;
   function Is_Empty (Obj : Root) return Boolean;

   type Derived is new Root and Writable with ...;
   procedure Read  (Obj : Derived; Data : out Integer);
   procedure Write (Obj : Derived; Data  : in  Integer);
end Pkg;
```

Package Pkg defines the root of derivation of a tagged type in which all descendants have the primitive operation Is_Empty. The package also has a derivation of Root that acquires the obligation of implementing all the primitives of interface Writable. Figure 2 presents the layout of an object of type Derived and its GNAT run-time structure.

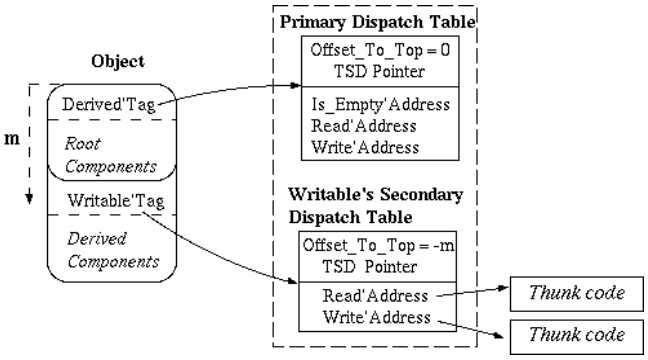

Figure 2 GNAT Object Layout

Each tagged type has one primary dispatch table, associated with its main root of derivation, plus one secondary dispatch table associated with each progenitor (a progenitor is one of the types given in the definition of a derived type other than the parent type ---AARM Annex N). In our example, each object of type Derived has one primary dispatch table plus one secondary dispatch table associated with the interface type Writable. Each dispatch table has a header containing the offset to the top and the pointer to the *Run-Time Type Specific Data* record (TSD). For a primary dispatch table, the Offset_To_Top component is always set to 0; for secondary dispatch tables the Offset_To_Top component holds the displacement to the top of the object from the object component containing the interface tag (in the figure the value of this offset is *m*). After the TSD component the dispatch tables have the table of pointers to primitive operations. In secondary dispatch tables, rather than containing direct pointers to the primitives associated with the interfaces, the dispatch table contain pointers to small fragments of code called *thunks*. These thunks are used to adjust the pointer to the base of the object during interface type conversions. For further information on the object layout and the GNAT run-time structures associated with interface types read [6, 7, 8]).

In order to present the new output associated with the construction of the dispatch table of type Derived let us compile the above Ada example using switch –gnatD:

```
Derived__ID :                                          -1-
  aliased constant string := "PKG.DERIVED";

Derived__Ifaces :                                      -2-
  aliased constant Interface_Data (1) :=
    (Num_Ifaces    => 1,
     Ifaces_Tables => (Derived__Writable_Tag));

Derived__TSD :                                         -3-
  aliased constant Type_Specific_Data (Idepth => 1) :=
    (Idepth              => 1,
     Access_Level        => 0,
     Expanded_Name       => Derived__ID'Address,
     External_Tag        => Derived__ID'Address,
     HT_Link             => null,
     Transportable       => False,
     RC_Offset           => 0,
     Interfaces_Table    => Derived_Ifaces'Address,
     SSD                 => null,
     Tags_Table          => (Derived_Tag, Root_Tag));

Derived__Predef_Prims :                                -4-
  aliased constant Address_Array (1 .. 10) :=
    (1  => Derived__Size'Address,
     ...
     10 => Root__DF'Address);

Derived__DT :                                          -5-
  aliased constant Dispatch_Table (Num_Prims => 3) :=
    (Num_prims      => 3,
     Signature      => Primary_DT,
     Tag_kind       => TK_tagged,
     Predef_prims   => Derived__Predef_Prims'Address,
     Offset_to_top  => 0,
     TSD            => Derived__TSD'Address,
     Prims_Ptr => (
       1 => Is_Empty'Address,
       2 => Read'Address,
       3 => Write'Address));

Derived__Tag :                                         -6-
  constant Tag := Derived__DT.Prims_Ptr'Address;

Register_Tag (Derived__Tag);                           -7-
```

At -1- we see the declaration of an object containing the external tag of Derived; at -2- we find the declaration and initialization of a table containing the tags of all the interfaces covered by Derived (in this example, just one); at -3- we have the Run-Time Type Specific Data record of Derived; at -4- we see the dispatch table of its predefined primitives; at -5- we see the primary dispatch table associated with Derived; at -6- we find the declaration of the Tag associated with this primary dispatch table (a copy of this tag will be saved in the _Tag component of objects of type Derived during their initialization); finally at -7- we find the code that registers the tags in the run-time (required to support the Internal_Tag service of standard Ada package Ada.Tags). For further information on the contents of each component see the documentation available in the source of *a-tags.ads*.

The expansion of dispatching calls makes use of the tag of the object and the compile-time known position of the target primitive to index the Prims_Ptr element containing the pointer to the target primitive. That is, considering the following example, in the commented line we see the expansion of the dispatching call to Is_Empty.

```
function Dispatch_Test (Obj : Root'Class)
  return Boolean is
begin
   return Obj.Is_Empty;
   -- Expanded into: return obj._Tag (1).all (obj);
end Dispatch_Test;
```

Source-based tools can use this new output to verify the correct construction of the dispatch table; they should check the subprograms referenced in the aggregates that initialize the dispatch table associated with predefined primitives (*Predef_Prims*) and the dispatch table containing the user-defined primitives (*Prims_Ptr*). For this purpose the compiler generates unique names for all subprograms found in the sources (including overloaded subprograms).

4.2 Static Allocation of Dispatch Tables

Another enhancement of the GNAT compiler is the improvement of its code generation to statically allocate dispatch tables associated with tagged types defined at the library level. In order to present it let us see the assembly code generated by GNAT when compiling the previous example for i86 architectures. For this purpose we compile our example using two additional switches (*-fverbose-asm* and *-save-temps*). The following fragment of assembly code corresponds to the declaration and initialization of the dispatch table containing the predefined primitives (object declaration found at -4- of previous Section):

```
pkg__derived_dt:
        .long pkg___size__2
        .long pkg___alignment__2
        .long pkg__derivedSR
        .long pkg__derivedSW
        .long pkg__derivedSI
        .long pkg__derivedSO
        .long pkg__Oeq__2
        .long pkg___assign__2
        .long pkg__rootDA
        .long pkg__rootDF
```

As the reader can see, the compiler generates external symbols for the table entries, rather than relying on the generation of run-time code to initialize table entries with addresses of code. For certification purposes, this is a major improvement in the code generation; previous versions of the compiler declare dispatch tables as un-initialized objects that are initialized during the elaboration of the package by means of additional assignments generated by

the compiler. Combined with GNAT specific pragmas this new feature allows placement of dispatch tables in ROM or in OS-guarded read-only memory.

4.3 Transformation of dynamic dispatching call into case-statement

Another concern for certifying OO code in HI software is the compiler support for Structural Coverage Analysis tools. The DO-178B stablishes three kinds of coverage requirements: Level C specifies *Statement Coverage*, which requires every statement in the program to have been invoked at least once. Level B specifies *Decision Coverage*, which requires every point of entry and exit in the program has been invoked at least once and every decision in the program has taken on all possible outcomes at least once. Finally, level A requires *Modified Condition/Decision Condition* (MCDC) testing, which involves testing all the permutations of conditions involving several logic operators. Dynamic dispatch complicates flow analysis of coverage requirements because reading the sources is it unclear which method in the inheritance hierarchy will be called [3, Section 2.2.3.1.1].

In order to help certifying Level A software with Ada, we are enhancing GNAT to expand dispatching calls into the equivalent case statements [2, 4]. The key point of this project is the following observation: although during the writing of any particular component of the program the final set of possible destinations of a dispatching call is unknown, this set is well known at link-time (we assume that a static linking step produces the executable for a given program). Therefore, instead of generating the usual transformation for a dispatching call, at the point of the call the GNAT compiler will generate the following code:

```
R := Obj.Is_Empty;
-- Expanded into: R := Find_Method (Obj, Obj'Tag);
```

The post-processing part of the code transformation is performed at bind-time. This involves generating the body for routine Find_Method which implements dynamic binding with an explicit case statement as shown bellow:

```
function Find_Method
  (Obj : Root'Class; The_Tag : Positive) return Boolean is
begin
   case The_Tag is
      when Root'Tag =>
         return Root (Object).Is_Empty;
      when Derived'Tag =>
         return Derived (Object).Is_Empty;
      ...
   end case;
end Find_Method;
```

Here the calls are not dispatching since the Object is converted to its actual subtype. The set of possible cases is complete since such transformation is done over the entire program.

The following implementation model is underway: a compiler option prevents the dispatching expansion described earlier, and a separate switch forces the binder to generate the source code for the case statements. This is legal Ada source code, which is therefore fully processable by standard tools, including the debugger and certification tools.

5 Conclusions

Ada is clearly a safe and efficient vehicle to create certifiable systems. It has been used successfully in many major aeronautics projects (Boeing 777, A340, and more recently Boeing 787, A380 and A400M). In the recent years Ada has evolved to fulfill the requirements of modern software industry incorporating object-oriented features into its original type model. The Ada 95 standard added to Ada tagged types, single inheritance, polymorphism, and dynamic dispatching. The latest revision of the language, informally known as Ada 2005 [11], adds multiple inheritance of abstract interface types and numerous other object-oriented programming idioms.

A preliminary version of the incoming DO-178C standard for avionics provides a comprehensive analysis on safety concerns associated with OO techniques in the context of DO-178B. Such document states that dispatching calls (the technique commonly used by most compilers for OO languages) is clearly unacceptable in this context. In this paper we have presented some enhancement projects of the GNAT technology that will help the industry to take advangage of the full benefits of the OO techniques with Ada without the inconveniences associated with dynamic dispatching, namely:

- **Dispatch table visualization**. Enhancement that modifies the compiler to make the initialization of the dispatch tables visible at the source level. In addition, the code generated by the compiler will be also visualized during debugging using another compiler switch. This project gives support to DO-178B traceability requirements.
- **Static allocation of dispatch tables**. Enhancement that improves the code generation of the compiler to allow the static allocation of dispatch tables associated with tagged types defined at the library level. This project will allow the placement of the dispatch tables in ROM or in OS-guarded read-only memory.
- **Translation of dispatching call into case-statement**. Enhancement that modifies the compiler to expand dispatching calls into the equivalent case statements. This project gives support to the structural coverage analysis and verification for level A systems as dictated by DO-178B.
- **Stack analysis tool**. This enhancement is already finished, and the gnatstack tool is currently part of the GNAT Pro toolset [1].

Since the emergence of the DO-178B standard, Ada has been one of the few languages of choice for the construction of HI systems. We expect that these enhancement projects to the GNAT technology will help Ada to keep this leadership.

Acknowledgements

Most of this work was done during a six-month visit to the NYU Courant Institute funded by the Spanish Minister of Education and Science under project PR2006-0356.

I give special thanks to professor Edmond Schonberg for his continuous help and support, and Professor Robert Dewar not only for the technical discussions but also for his hospitality. I also acknowledge the contributions of Cyrille Comar, Franco Gasperoni, Richard Kenner, and Eric Botcazou that helped me to go ahead with this work.

References

[1] E. Botcazou, C. Comar, and O. Hainque (2005), *Compile-time stack requirements analysis with GCC*, June, 2005. Available at: http://www.adacore.com/2005/06/01/compile-time-stack-requirements-analysis-with-gcc/

[2] C. Comar, R. Dewar, and G. Dismukes (2006), *Certification & Ob ject Orientation: The New Ada Answer*, March, 2006. Available at: http://www.adacore.com/2006/03/08/certification-object-orientation-the-new-ada-answer/

[3] FAA (2004), *Handbook for Object-Oriented Technology in Aviation* (OOTiA), October 26, 2004. Available at: http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/oot/

[4] F. Gasperoni (2006), *Safety, Security, and Object-Oriented Programming,* March, 2006. Available at: http://www.adacore.com/2006/03/30/safety-security-and-object-oriented-programming/

[5] J. Gosling, B. Joy, G. Steele, and G. Bracha (2000), *The Java Language Specification*, 2nd edition, Addison-Wesley.

[6] J. Miranda, E. Schonberg, and G. Dismukes (2005), *The Implementation of Ada 2005 Interface Types in the GNAT Compiler*, 10th International Conference on Reliable Software Technologies, Ada-Europe 2005, LNCS 3555, pp. 208–219, Springer-Verlag.

[7] J. Miranda, Schonberg E., and G. Dismukes (2006), *Abstract Interface Types in GNAT: Conversions, Discriminants, and C++*, 11th International Conference on Reliable Software Technologies, Ada-Europe 2006, LNCS 4006, pp. 179–190, Springer-Verlag.

[8] J. Miranda, Schonberg E., and H. Kirtchev (2005), *The Implementation of Ada 2005 Synchronized Interfaces in the GNAT Compiler*, Proceedings of the 2005 annual ACM SIGAda International Conference on Ada, pp. 41–48.

[9] RTCA (1992), *Software Consideration in Airborne Systems and Equipment Certification*, RTCA/DO-178B, December, 1992.

[10] RTCA (2001), *Final report for clarification of DO-178B: Software Consideration in Airborne Systems and Equipment Certification*, RTCA/DO-248B, October, 2001.

[11] S. T. Taft, R. A. Duff, R. L. Brukardt, E. Plödereder, P. Leroy (eds) (2007), *Ada 2005 Reference Manual: Language and Standard Libraries International Standard ISO/IEC 8652:1995(E) with Technical Corrigendum 1 and Ammendment 1*, LNCS 4348, Springer-Verlag. ISBN: 3-540-69335-1.

[12] VEROCEL (2006), *VerOLink: Verify Object Linking*, http://www.verocel.com/verolink.htm.

SAMATE and Evaluating Static Analysis Tools

Paul E. Black

National Institute of Standards and Technology, 100 Bureau Drive Stop 8970, Gaithersburg, MD 20899; email: paul.black@nist.gov

Abstract

We give some background on the Software Assurance Metrics And Tool Evaluation (SAMATE) project and our decision to work on static source code security analyzers. We give our experience bringing government, vendors, and users together to develop a specification and tests to evaluate such analyzers. We also present preliminary results of our study on whether such tools reduce vulnerabilities in practice.

Keywords: software assurance, source code, static analysis, tool testing.

1 Introduction

The Software Assurance Metrics And Tool Evaluation, or SAMATE, project [11] at the US National Institute of Standards and Technology (NIST) focuses on one aspect of reliable software: software assurance, particularly security assurance. That is, how can we gain assurance that software is secure enough for its intended use? The SAMATE project seeks to help develop standard evaluation measures and methods for software assurance.

High levels of quality and security cannot be "tested into" software. Such attributes must be built into software from the beginning, starting with requirements and choice of environment. Preventing flaws is far more cost-effective and dependable than trying to remove them. But what if the system being designed includes commercial, off-the-shelf (COTS) packages? How can a contractor thoroughly audit or check large packages from subcontractors? Legacy code may need reviews before being used in a new environment or for newly discovered threats. Also for quality assurance, one needs to know what kinds of flaws a current development process might leave or whether a new method yields better quality software. In all these cases, one must work with the code that is available.

Although SAMATE will eventually consider the impact of using better programming languages, such as Ada[1] or Eiffel, advanced software development approaches, and correct-by-construction techniques, we started with software metrics and understanding tools for checking software.

In the software realm, what can we do to increase software assurance? We can enable tool improvement and encourage wider use of tools. We will simultaneously urge use of better environments, practices, and languages.

Some questions quickly spring to mind. If a tool gives no outstanding alarms for a system, how secure is the system, really? Is the new version of a tool better (pick your own definition) than the preceding version? How much better? Which tools find what flaws? To answer these questions, we must back up and try to make a comprehensive list of flaws that might occur and have a taxonomy of software security assurance tools and techniques which might be investigated.

Both tasks have proved far harder than first thought. The effort to list flaws led to Mitre's Common Weakness Enumeration (CWE) [5] effort. Although not complete, the SAMATE web site has the latest version of the taxonomy of software security assurance tools and techniques [14].

For clarity we quote some definitions from our Source Code Security Analysis Tool Functional Specification Version 1.0 [13]. It defines a *vulnerability* as "a property of system security requirements, design, implementation, or operation that could be accidentally triggered or intentionally exploited and result in a security failure. … If there was a security failure, there must have been a vulnerability." It continues, "a vulnerability is the result of one or more *weaknesses* in requirements, design, implementation, or operation." We use the term "weakness" to emphasize that without the entire context, one cannot truly conclude that a problem may occur. Some other code or part of the system may prevent the weakness from being exploited.

Why we started with static source code analyzers

Higher level representations, such as requirements or use cases, are better places to prevent flaws. But we did not find any area mature enough for standardization. Also roughly half of all security weaknesses are introduced during coding [7], so improvement after high level design may be helpful. Unlike binary or byte code, source code is largely human-readable. Also there are many tools that work with source code. For these reasons, source code seems a good place to begin.

Testing and static analysis complement each other. Static analysis is less feasible without source code. It may be impossible, say, in testing embedded systems or remote testing of Internet services. On the other hand, one cannot test an incomplete program, while static analysis might be feasible. More importantly, testing is unlikely to uncover very special cases, for instance, granting access when the user name is "matahari".

[1] Any commercial product mentioned is for information only. It does not imply recommendation or endorsement by NIST nor does it imply that the products mentioned are necessarily the best available for the purpose.

We started with static source code security analysis tools since they have a potential of higher assurance. Chess and McGraw [3] give an excellent short introduction to the use of static analysis for software assurance. Note that when we use the term "tool", we are actually referring to a set of functionalities. That is, any program that can statically analyze source code is in the class. It need not be a stand-alone tool that analyzes source code and does nothing else.

Our first funding, from the US Department of Homeland Security, was to develop tests for tools. After talking with vendors, we decided we could increase adoption by developing basic tests.

One reason software developers do not adopt a tool is they are not sure whether any tool of the sort is really helpful and whether a particular tool is actually broadly useful. Some demonstrations are set up to give the appearance of great performance, when the breadth or depth or power is disappointing in practice. A standard developed by NIST would help assure software developers that source code security analyzers are useful. In addition, when a tool satisfies the standard, the user has some assurance that it has adequate coverage. Speeding the adoption of these tools can increase vendors' sales: the market for source code analyzers can grow a lot before it is saturated.

2 A specification for static source code security analyzers

NIST is a non-regulatory agency. This means we cannot mandate the use of our standards or tests. We must "sell" our results. One step to acceptable tests is a widely accepted specification of what such a tool should do.

We need the cooperation of vendors and developers of source code security analyzers to efficiently succeed. They have experience with what users, or at least their customers, need and want from such analyzers. They also have experience with what is practical, having written production analyzers. Some of the best researchers in the field have been the primary agents in developing these commercial tools. Finally we want our standard to be an endorsement that vendors will seek, rather than something to be forced on them. How can we position this effort to help vendors and consumers?

We looked for available source code security analyzers on the web, in published articles, and by personal contact. We the updated collection of tools is on-line at [12]. We organized several workshops and conference sessions on source code security analyzers. Vendors were willing to attend, and we discussed possible approaches and goals with them. To build a consensus, we established a mailing list, where we discussed facets, and made drafts of the specification available for public review and comment.

Informally a static source code security analyzer (1) examines source to (2) detect and report weaknesses that can lead to security vulnerabilities [13]. Tools that examine other artifacts, like requirements, byte code or binary, and tools that dynamically execute code are not included. Again, when we use the term "tool", we mean a set of capabilities of a tool.

Before continuing, we must decide the purpose of the specification and tests. It would be nice if they could serve as a metric to completely characterize the capabilities of a tool, but that is not possible, even in theory. A bit more practical specification could establish a lofty goal whose satisfaction ensures the user got the level of security checking needed. But since different users have different security needs, this is complicated. A specification could settle for a recommended standard, like due diligence. Even here, we have little objective evidence to establish such a level.

We chose to work for a minimum standard to begin with. A minimum standard would reduce argument about how high a level is right and exactly what should be required. Although insufficient for, say, setting recommended practices, a minimum standard opens the way for a higher standard.

What exactly should a source code analyzer do?

In more detail, such an analyzer should find weaknesses and report their class and location. The weakness class corresponds to CWE entries. Many tools also report conditions that may expose the weakness, data or control flow related to it, more information about that class of weakness including examples of how to fix it, the certainty that the weakness is a vulnerability (not a false alarm), or some rating of the severity or ease of exploit.

Optionally a tool should produce a report that could be used by other tools. For practical use in repeated runs, there must be some mechanism to suppress reports of weaknesses judged to be false alarms or otherwise to be subsequently ignored.

False positives are a critical factor. Conceptually, static analysis tools compute a model of a program. They then analyze the model for certain properties. Since static analysis problems are undecidable in general, either the computed model is approximate or the analysis is approximate. Due to these approximations, tools may miss weaknesses (false negatives) or report correct code as having a weakness (false positives). To be adopted a tool must "have an acceptably low false positive rate" [13].

Nowhere in the specification is a rate given. One reason is that a rate that is acceptable for one application or development situation may not be acceptable for another. Why then bother having the requirement? It is a NIST practice to only test items in the specification. It would be "poor sportsmanship" to test for a false positive rate without a written requirement. With more research we hope to be able to give acceptable rates, at least for some situations.

Other issues for a specification

The expressive power of programming languages makes analysis even harder. Analysis routines must be explicitly developed to handle coding complexities, such as loops, conditional control flow, arrays, different variable types,

interprocedural function calls, and aliasing. In practice no tool handles all possible code constructs. To assure the user that a tool handles some, the specification also requires that weaknesses be found in the presence of a set of coding complexities.

No tool checks for all weaknesses in the CWE. Some are hard to define, like leftover debug code (CWE ID 489). With over 500 weaknesses tool developers concentrate on a relatively small set of frequent or severe weaknesses and put effort saved into improving analysis and user aids.

We chose a "minimum" set of weaknesses: those that are most common or occur most often, most easily exploited, and are caught by existing tools.

To be as flexible as possible the specification should explicitly refer to the subset of weaknesses that a tool purports to catch. However this causes severe problems in developing test material that also covers coding complexities.

We need to check that all weaknesses (in our "minimum" set) are caught in the presence of all coding complexities. The naive test suite would have every weakness in the presence of every coding complexity. This would be thousands of tests, complicating the creation and running of the test suite. Since we expect the same analysis modules to handle coding complexities for all weaknesses, we believe having each weakness in the presence of a few coding complexities, where every coding complexity occurs at least once, has substantially the same testing power. This test suite has less than 100 test cases.

Allowing for a subset of weaknesses presents challenges. In the extreme, what if a tool only purports to catch one weakness? The three or four test cases from the test suite will not exercise all coding complexities. We see a number of possibilities.

We could go back to having thousands of cases, so any weakness also exercises all coding complexities, but it would be unwieldy. One option is to develop a generator to create a custom test suite with the coding complexities distributed throughout as many or as few weaknesses as desired. Another possibility is to prepare adequate test suites as needed, in hopes that a limited amount of work would address real needs. Trusting that most tools cover a minimum set, we discarded the allowance for subsets from the specification. But we now find that unrealistic.

As part of the result of a study of tools Britton [2] reported, "84 percent of the vulnerabilities found were identified by one tool and one tool alone". Rutar, Almazan, and Foster [9] concluded that tools for finding bugs in Java do not overlap much in what they catch. In consolidating weakness classes found by five tools Martin [6] reported little overlap: few weaknesses were even caught by two tools. Currently the best approach, that is lowest false positive and highest identification rates over many weaknesses, is to use two or more tools as a combined metatool.

What are the attributes of test cases? Small cases separate the question of "can this be detected" from "how scalable is the tool". On the other hand, large programs allow examination of speed and maximum size and exercise a tool in a more realistic situation. Having one weakness in each test case makes analysis of the result easier, but having multiple weaknesses in one program should be more challenging. It is straightforward to write code with known weaknesses, whereas extracting examples from production code disarms the objection that it is unrealistic. Trying to find an instance of a weakness and extracting a slice of code could take excessive amounts of time. Even with the slice, we would have to secure permission to make it publicly available.

Currently our test cases are very small, purpose written code with one weakness per case. For measuring the false positive rate we also have cases without weaknesses or in which weaknesses have been fixed.

Sharing example code

While researching source code security analyzers, we found it difficult to get a corpus of code with known weaknesses. Although academicians develop them for research, companies have some for testing, and evaluators assemble them, few were available. We felt a single repository of such could be helpful to the entire community. Not only would it provide a place for us to keep and publish our test cases, it would allow people to share the work they've done.

The SAMATE Reference Dataset (SRD) [10] is an on-line, publicly available repository of thousands of samples of flawed software. Each test case consists of one or more files. Test cases may consist of source code, byte code, binaries, requirements, or other artifacts.

Each test case may have explanatory information associated with it, for instance, the author or contributor, the date submitted, language, which flaw(s) it exhibits, and a description. In addition, test cases may have directions on how to compile and link source code, input that triggers the flaw, or expected output. Registered users can submit test cases and add comments to any test case.

For historical stability, the content of test cases will never be updated. If the code in a test case needs to be fixed or improved, a new test case will be added, and the status of the existing test case will be changed to "deprecated". Deprecated status advises against using the case for new work. This way, a test report referring to a certain test suite can be rerun exactly, even years later.

Methods to minimize test evasion

A fixed, public test set allows for various abuses. A tool developer may write special-purpose code to get the right result for a very special case. This diverts effort from general improvements and incorrectly raises ratings. More simply a developer may add code to recognize the case, perhaps the name and size of the file, and hard-code the right result.

We see several ways to minimize such distractions. Easiest is to keep the test set secret. The test set would only be shared with parties trusted to keep it private. Practically a public version would be needed to allow others to examine and critique the tests and to allow vendors to practice. Even with a public version, it might be difficult to convince tool developers that the private tests are fair and reasonable.

Another possibility is to develop a test generator that creates unique test sets on demand. The challenge with this approach is to ensure that every test set generated is similar in its testing power. Adding code from production applications or getting large pieces of code would be hard.

We are writing an obfuscator to discourage developers from having their program recognize tests and return predetermined answers. At a minimum the obfuscator must change source code file names. The next level is to change comments and names in the source code, such as variable and function names. Although very hard in general, the obfuscator only needs to work on the test suite.

Since most source code analyzers already have powerful abstraction capabilities, they could store signatures of abstract syntax trees and return prepared responses when one is recognized. To foil this, the obfuscator could insert benign code or rearrange existing code. Rather than requiring full code rewriting capabilities in the obfuscator, test cases could be in some preprocessed form or have "hints" stored. In this case, a macro processor could generate many different versions of the test set. Developers still might be tempted to add special case algorithms to improve results.

3 Do tools really help?

One can think of several potential problems with the use of such tools in practice. A tool may report many weaknesses, but miss the small number of serious weaknesses that really affect security. If a user takes a mechanical approach to fixing weaknesses reported by tools, programmers may not think as much about the program logic and miss more serious vulnerabilities. Also, the developer may spend time correcting unimportant weaknesses reported, making other mistakes in the process and not having as much time for harder security challenges. Recognizing such problems, Dawson Engler [4] articulated the question: "Do static source code analysis tools really help?"

Funded by the US Department of Homeland Security, Coverity, in collaboration with Stanford University, has analyzed over 50 open-source projects since March 2006 [1]. As an example, they reported over 600 defects in Firefox and 98 defects in Python. At least one security vulnerability was detected: CVE-2006-0745. Others have similar, although smaller, scans. Maintainers may use these reports to fix previously unknown vulnerabilities. By studying these, we may be able to support or refute Engler's question.

We are examining the history of reported vulnerabilities for the projects scanned by Coverity. We use reported vulnerabilities as a surrogate measure for actual vulnerabilities. The null hypothesis is that there is no change in the number of reported vulnerabilities after the start of scanning. We give preliminary results we have for one project, MySQL.

Coverity scanned MySQL version 4.1.8 in early 2005. Version 4.1.10, released 15 February 2005, contained fixes based on Coverity reports. Figure 1 compares vulnerabilities discovered in version 4.1.10 or later versions with vulnerabilities discovered before the 15 February release. "Discovery" means it was reported in the National Vulnerability Database (NVD) [8].

Red bars, on the right, are vulnerabilities discovered in version 4.1.10 or later. They are grouped by discovery date. As the discovery date, we used the earlier of the discovery date in the NVD and in the SecurityFocus database [15]. Our data covers 21 months after the release of version 4.1.10. The light blue bars, on the left, are vulnerabilities discovered before the release. We began 21 months before the release, that is May 2003. Vulnerabilities discovered after 15 February 2005 that were only present in versions before 4.1.10 were not used.

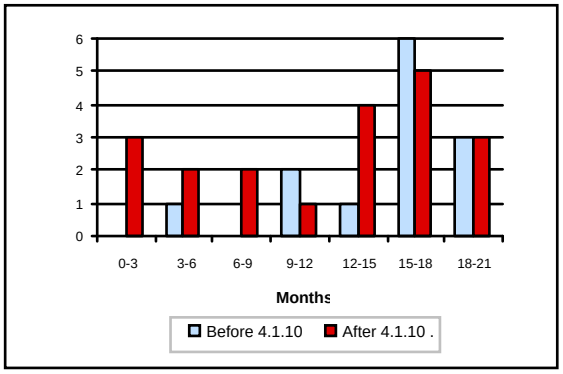

Figure 1 MySQL vulnerabilities before and after 4.1.10

The data is insufficient to draw any conclusions. We are trying to take confounding factors into account, and we are analyzing similar data from other projects to accumulate a statistically meaningful set.

4 Future directions

We are planning several studies to answer questions such as the following. How does one tool's assessment correlate with another tool's assessment? What is the subject of a metric, that is, does it apply to the algorithm, an implementation, or an execution trace?

We are currently working on specifications and tests for web application scanners. The next class of tool we will work on is binary analyzer. We are also guiding efforts to formalize descriptions of weaknesses. Although formal description will have many uses in the long term, our immediate application is a test case generator. The generator uses the descriptions to produce example code.

We are looking for collaborations. In particular, we need a few more people to serve on our technical advisory panel, which meets once or twice a year to suggest where we might be of most help in the future. We also seek

participants in focus groups to review specifications and test plans for classes of tools.

In the long term we plan to go beyond tools, especially checking tools. Society must move beyond a catch-and-patch approach. We will help develop metrics to gauge more secure languages, good processes, environments, etc. We want to help demonstrate what really improves software security.

References

[1] *Accelerating Open Source Quality*, http://scan.coverity.com/ (Accessed 21 May 2007).

[2] Peter A. Buxbaum (2007), *All for one, but not one for all*, Government Computer News, 26(6), 19 March. Available at http://www.gcn.com/print/26_06/43320-1.html (Accessed 22 May 2007).

[3] Brian Chess and Gary McGraw (2004), *Static Analysis for Security*, Security and Privacy Magazine, IEEE, 2(6), pp 76-79.

[4] Andy Chou, Ben Chelf, Seth Hallem, Charles Henri-Gros, Bryan Fulton, Ted Unangst, Chris Zak, Dawson Engler, *Weird things that surprise academics trying to commercialize a static checking tool*, http://www.stanford.edu/~engler/spin05-coverity.pdf (Accessed 21 May 2007).

[5] *Common Weakness Enumeration*, The MITRE Corporation, http://cwe.mitre.org/ (Accessed 21 May 2007).

[6] Robert A. Martin (2007), *Making Security Measurable*, Providing Assurance in the Software Lifecycle DHS-DoD Software Assurance Forum, Fair Lakes, Virginia.

[7] Gary McGraw (2006), *Software Security*, Addison-Wesley.

[8] *National Vulnerability Database*, National Institute of Standards and Technology, http://nvd.nist.gov/ (Accessed 21 May 2007).

[9] Nick Rutar, Christian B. Almazan, and Jeffrey S. Foster (2004), *A Comparison of Bug Finding Tools for Java*, 15th IEEE International Symposium on Software Reliability Engineering, IEEE Computer Society, pp 245-256. Available at http://www.cs.umd.edu/~jfoster/papers/issre04.pdf (Accessed 21 May 2007).

[10] *SAMATE Reference Dataset*, National Institute of Standards and Technology, http://samate.nist.gov/ SRD/ (Accessed 20 May 2007).

[11] *Software Assurance Metrics And Tool Evaluation (SAMATE) project*, National Institute of Standards and Technology, http://samate.nist.gov/ (Accessed 20 May 2007).

[12] *Source Code Security Analysis*, National Institute of Standards and Technology, http://samate.nist.gov/index.php/Source_Code_Security_Analysis (Accessed 24 May 2007).

[13] *Source Code Security Analysis Tool Functional Specification Version 1.0*, National Institute of Standards and Technology, Special Publication 500-268, May 2007. Available at http://samate.nist.gov/docs/source_code_security_analysis_spec_SP500-268.pdf (Accessed 24 May 2007).

[14] *Tool Taxonomy*, National Institute of Standards and Technology, http://samate.nist.gov/index.php/Tool_Taxonomy (Accessed 25 May 2007).

[15] *Vulnerabilities*, SecurityFocus, http://www.securityfocus.com/vulnerabilities (Accessed 25 May 2007).

Identifying Opportunities for Worst-Case Execution Time Reduction in an Avionics System

Guillem Bernat, Robert Davis, Nick Merriam, John Tuffen

Rapita Systems Ltd. IT Centre, York Science Park. YO10 5DG UK.
Tel: +44 1904 567747; Email: bernat@rapitasystems.com

Andrew Gardner, Michael Bennett, Dean Armstrong

Hawk Mission Systems, BAE Systems. Brough. HU15 1EQ.

Abstract

This paper describes the results of a study that identified opportunities for worst-case execution time reduction in the Operational Flight Program (OFP) software of BAE Systems' Hawk Mission Computer. The RapiTime toolset was used to provide the execution time analysis information required to target optimizations where they would be most effective. Potential optimizations were identified for worst-case hotspots at three levels: design level, sub-program level and low-level. These hotspots accounted for only 1.2% of the lines of code but contributed 29% of the overall execution time. Focused optimizations on these hotspots resulted in a 23% reduction of the overall execution time for the analysed code.

Keywords: Worst-case execution time analysis, WCET, real-time, Ada, Avionics, performance measurement.

1 Introduction

In a real-time system, it is important to guarantee both functional and non-functional requirements, in particular timing correctness. Functional verification is a well understood process that includes requirements capture, design, implementation, review and testing. However, the process for timing verification is less well understood. Current trends towards more complex software and more advanced hardware have resulted in the need to spend significant time and effort in understanding, verifying and improving the timing behaviour of systems.

One such complex system is the Operational Flight Program for the BAE Systems' Hawk Mission Computer[1]. The Operational Flight Program software is written in Ada and consists of hundreds of thousands of lines of code divided into 25 partitions, themselves divided into tasks, executed in a cyclic schedule. In 2006, the current system was running close to capacity, in terms of available execution time. In order to provide capacity for new functionality, an internal activity was commenced to identify optimization opportunities that would reduce the worst-case execution time of the system by at least 10%; thus avoiding the need for an expensive hardware upgrade.

Figure 1 Hawk fast jet trainer

Manual identification of optimization opportunities in such a large system is a daunting task. In this case, the system was developed over a number of years by a large team of engineers. Its sheer size and complexity makes it difficult if not impossible for a single engineer to gain an in depth understanding of the entire system. Further, there was no clear information on which components actually contributed to the overall worst-case execution time.

Initial efforts at understanding the timing behaviour of the system were based on determining the execution time of each partition via *high water marks* measured on the target microprocessor.

A typical situation was that painstaking optimization of a sub-program would result in unit tests showing a significant reduction in execution time whilst making little or no impact on the overall high water mark. In contrast, simpler more minor optimizations could sometimes have a significant impact, reducing the high water mark readings. This occurred when, in the first case, the code was not actually on the worst-case path, and in the second case, when the sub-program was both on the worst-case path and called a large number of times on that path.

Initially, there were no mechanisms in place to identify which sections of code were on the worst-case path, thus the selection of which sub-programs to optimize was effectively an educated guess.

[1] Hawk is a fast jet trainer, famously flown by the Red Arrows display team [1].

Note that conventional profiling mechanisms are not particularly useful in the solution of this problem as they identify code that contributes most to the average execution time, which may be completely different from the code that is on the worst-case path and contributes most to the worst-case execution time.

Rapita Systems, together with the Hawk Integration Team, investigated how the problem of identifying the correct targets for optimization could be solved using the RapiTime worst-case execution time and performance analysis toolset. The study also aimed at evaluating the capabilities of the RapiTime toolset to cope with very large Ada programs, and its ability to summarise execution time data so that optimization opportunities could be easily identified and prioritised.

Using RapiTime, the joint project team made up of Hawk Integration Team and Rapita Systems engineers was able to successfully analyse the selected subset of the system. RapiTime was used to identify the small number of sub-programs that contributed heavily to the worst-case execution time. This code was inspected and, using the worst-case hotspot information provided by RapiTime, key code constructs targeted for optimization. These optimizations were classified as: low-level, sub-program level and design level. The best candidates for optimization were prototyped and the new system analysed to verify the effectiveness of the changes. The results of these optimizations are reported in Section 5.

The remainder of the paper is organised as follows: Section 2 describes in more detail the system under study; the Operational Flight Program of the Hawk Mission Computer. Section 3 provides a brief overview of the RapiTime toolset. Section 4 gives a classification of optimization opportunities, with examples of constructs found at each level. Section 5 reports the main results of the study and finally, Section 6 provides a summary and conclusions.

2 Hawk Operational Flight Program

The system under analysis is a subset of the Mission Computer of the HAWK aircraft. The Mission Computer provides the graphics for all six cockpit display panels and head-up displays amongst other functionality.

2.1 Architecture

The software for the Operational Flight Program (OFP) is written in Spark Ada and comprises several hundred thousand lines of source code. The OFP is divided up into 25 partitions executed as part of a cyclic schedule.

The system runs on a microprocessor board based on the PowerPC MPC7410 running at 500 MHz. This processor has a significant number of complex hardware features. It includes a two level cache: separate data and instruction level-1 caches of 32 Kbytes each, with pseudo random replacement policy, and a 2 Mbytes integrated level-2 cache. It also has a branch prediction unit, a 9-stage pipeline, multiple instruction fetches per cycle, a floating-point unit, two integer units and a performance counter unit. The board has 512 Mbytes of RAM.

2.2 Previous approach to timing analysis

Prior to the study, the method used to obtain timing information about the Hawk OFP software involved taking average and 'high water mark' measurements of the time each partition or task took to execute during testing or normal operation. High water marking was implemented by simply recording the time at the start and end of the partition (or any arbitrary section of code) on each execution and subtracting these two values to determine the execution time. This value was then compared to the largest value found so far and if greater, the new value was kept. At the end of execution these high water mark values could be examined.

With this process, no on-target code coverage information was available, so it was not possible to determine how much of the code was actually exercised by the tests. One potential risk was that the real worst-case execution time could be much longer than the high water mark value due to code that had not been exercised.

The high water mark approach had the further disadvantage that the large number of software components involved were unlikely to take their worst-case execution times together. Hence the high water mark times ran a significant risk of being optimistic i.e. less than the real worst-case time, even for the set of sub-programs that were fully exercised by the tests.

Unfortunately, often the first indication of a problem with the timing behaviour of the system was when it overran its timing budget during the latter stages of testing. At this point, manual intervention to discover which components were the main contributors to the overrun required additional effort and resources, resulting in potentially expensive and time-consuming delays.

The RapiTime toolset enabled a systematic, efficient and effective approach to be taken in investigating the timing behaviour of the system and identifying the worst-case hot-spots that were the major contributors to the overall execution time. The process of analysing the system using RapiTime is described in the next section, follow a brief overview of the RapiTime toolset.

3 RapiTime

Obtaining accurate information about the longest time a piece of software can take to run, termed the *worst-case execution time,* is key to ensuring that time constraints are met and that a real-time system operates correctly.

RapiTime [3] is an analysis toolset that provides a *unique* solution to the problem of determining worst-case execution times for complex software running on advanced microprocessors.

RapiTime uses an innovative combination of three techniques:

1. The best possible model of an advanced microprocessor is the microprocessor itself. RapiTime therefore uses *online* testing to measure the execution time of sub-paths between decision points in the code.
2. By contrast, *offline* static analysis is the best way to determine the overall structure of the code and the paths through it. RapiTime therefore uses path analysis techniques to build up a precise model of the overall code structure and determine which combinations of sub-paths form complete and feasible paths through the code.
3. Finally RapiTime combines the measurement and path analysis information in a way that accurately captures the execution time variation on individual paths due to hardware effects.

The RapiTime toolset can be used to:

- Determine worst-case execution times for each software component, from complex programs down to basic blocks of code.
- Identify code that is on the worst-case path.
- Provide detailed analysis of *worst-case hotspots* and their contribution to the overall worst-case execution time.
- Provide code-coverage metrics ensuring confidence in the analysis results.
- Generate Execution Time Profiles illustrating the variability in execution times due to hardware effects.

RapiTime not only computes maximum values of execution times, but also their full distribution (in a statistical sense), thus capturing the variability of execution times due to hardware effects. This analysis is based on state-of-the-art statistical methods for modelling statistical dependencies known as the theory of Copulas [2].

3.1 RapiTime analysis process

The RapiTime toolset integrates into the standard software build process. As part of the study, makefile scripts were modified to include the following extra steps required for RapiTime to analyse the system:

1. Build executables for analysis. A special build was produced that had the subsystem under analysis automatically instrumented as well as including a lightweight tracing library for the MPC7410.
2. Structural analysis. The make process was modified to include an extra step, allowing the RapiTime tools to extract the structure of the code. The structure was derived from analysis of the disassembled executable, capturing the transformations that the compiler introduced into the code.
3. Testing and trace generation. This stage involved running the application on the target microprocessor under a set of test scenarios, collecting the trace data and downloading it from the target. Several options exist to extract the timing data from the target. In this case, the standard debugger was used to download a memory dump of the area in memory where the trace data was stored.
4. Trace processing. RapiTime trace manipulation tools were used to extract timing traces from the memory dump, to filter out events of no interest, to compress the data, to fix timer wraparounds, and finally, to derive a set of measured execution time profiles for each sub-program, loop and basic block.
5. Worst-case execution time calculation and report generation. The final stage was the WCET calculation using the measured data from individual sub-paths and structural information about the code. Additional annotations were used at this stage to guide the calculation process. The results were formatted as a set of easy to navigate reports.

The information in the RapiTime reports was used to identify those sub-programs that contributed most to the worst-case execution time and thus select the most promising opportunities for optimization. Opportunities for optimization were considered at three levels: design-level, sub-program level and low-level. These three categories are described in more detail in the next section.

4 WCET optimizations

Optimization is a compromise of several factors, in particular: time, space, readability, maintainability and effort. For example, some optimizations may lead to code structures that are very hard to maintain but result in a significant reduction in execution time. The key to any optimization strategy is to prioritise those optimizations where the minimum effort (and the minimum amount of compromise in other factors) is required to gain the maximum benefit in execution time reduction.

Profiling is not worst-case. Unlike conventional code profiling techniques, RapiTime identifies the worst-case hotspots in a program from the point of view of execution time. That is the lines of code that contribute the most to the worst-case execution time. Conventional profiling techniques identify the lines of code that execute the most *on average*, which is very different. For example, in the following code:

```
if rare_condition_of_error then
  long_computation_to_fix_the_error;
else
  short_normal_operations;
end if;
```

A profiler would indicate that most of the time is spent performing the short_normal_operation, missing the fact that in the worst-case, the path to follow is through long_computation_to_fix_the_error. Any optimization

performed on the else branch would have no impact at all on the overall worst-case execution time.

For example, the following code is optimized for the average case:

```
if most_of_the_time then
  short_execution_time;
elsif less_regularly then
  medium_execution_time;
elsif very_infrequently then
  long_execution_time;
end if;
```

However, in the worst case the code needs to do the three tests, an optimization for the worst-case would instead be:

```
if very_infrequently then
  long_execution_time;
elsif most_of_the_time then
  short_execution_time;
elseif less_regularly then
  medium_execution_time;
end if;
```

In this case, only one test is done on the worst-case path.

On a similar note, deciding with bit of code to lock in the cache may also be different for worst-case optimization than for average case optimization. For example in the previous example if long_execution_time took a very long time but actually used few cache lines, it would be a good candidate to be locked in cache.

4.1 Level of focus

A key focus of the optimization process is to identify the level at which to perform the optimization. Optimizations can be classified at three levels: design-level, sub-program-level and low-level.

Design-level optimizations

Optimizations at the design-level, as the name suggests, refer to changes in the overall design of the system. These optimizations may involve changes in the way in which software components communicate, changes in APIs, and changes in how components are structured and subdivided. For example, use of Ada generics may lead to longer execution times as some compilers fully inline the code, therefore missing significant benefits of instruction cache. Changing the architecture of the program to use less generic components and re-usable APIs has other consequences related to ability of the compiler to do constraint checking.

Analysis at this level is usually difficult and expensive as it may require changes to the overall system design, which can have significant impact on implementation and testing. However, very significant improvements in execution time can be achieved by changes at the design-level.

Sub-program-level optimizations

Optimizations at the sub-program level focus on changes within a single sub-program (or a set of tightly coupled sub-programs) without changing the specification of those components. Examples of these optimizations are changing the complexity of an algorithm, for example from an $O(n^2)$ to an $O(n \text{ Log } n)$ sort routine, changing an iterative process to one using lookup tables, loop unrolling, and avoiding making extra copies of data, therefore reducing memory footprint and the potential for cache misses.

Low-level optimizations

Low-level optimizations focus on the generated machine code. These optimizations aim to use the most efficient available machine instructions for performing particular tasks. For example, in some DSP processors, specific machine instructions exist to find the first-set bit or count the number of set bits in a word. These instructions are significantly faster than typical software implementations of the same functionality. Another example of machine dependencies is using non-native word sizes. This may result in significantly larger and slower code. A programmer who is not aware of this fact may miss an important opportunity for optimization.

Another important aspect is the nature of the generated code, especially relevant for Ada is the fact that a few lines of code can result in a very long execution time. For example:

```
type T is new integer;
type U is array ( 0 .. 10000) of Big_Record;
  …
  a,b : T;
  c,d : U;
  …
  a:=b; -- single copy of integer
  c:=d; -- can take a very long time to run!
```

The two last statements, although very similar at the source code level result in very different object code.

A particular aspect of importance at this level is the identification of the impact that compiler optimizations have on the code. For example, on modern processors with large caches and small memories, using compiler optimization for size can, counter-intuitively, result in better execution time performance than using compiler optimization for speed. This occurs when the bottleneck is actually fetching code from main memory, rather than the actual processing of those instructions.

4.2 RapiTime optimization process

RapiTime provides information on the percentage contribution of each sub-program to the overall execution time. This information is used to identify candidate sub-programs for optimization. The best candidates for optimization are then inspected. This involves studying both the Ada source code and in some cases the object code generated by the compiler.

Next, RapiTime is used to answer what-if questions about the effects of potential reductions in the execution time of these sub-programs. This shows that optimizing some candidate sub-programs would result in a commensurate reduction in the overall worst-case execution time; whilst for other sub-programs, optimization would bring little benefit as the worst-case path shifted to other code.

Of particular importance is the fact that even though a sub-program can be a worst-case hot-spot, its optimization may not necessarily lead to a significant reduction in the overall worst-case execution time if by optimizing that code, the worst case path switches to another path. For example

```
If some_condition then
  A; -- on worst-case path. Takes 10 ms
else
  B; -- not on worst-case path. Takes 9ms
end if;
```

In this example, reducing A by more than 1 ms, switches the worst-case path to the branch B, therefore both A and B need to be optimized together to reduce the worst-case execution time.

Quantification of the improvement

An important aspect of the optimization process is a final review examining the impact and consequences of the optimization process. This review quantifies the reduction in execution time, and also assesses the impact of code changes on portability, maintainability, code size, etc. Some optimizations may be rejected at this stage if they do not bring sufficient benefits to warrant for example non-portable or difficult to maintain code.

5 Main results of the study

This section describes the main results of the study. The target for phase 1 of the study (reported here) was to deliver a saving in the overall schedule, corresponding to 100 execution time units (ETUs)[2]. Achieving this reduction would put the project well on track to achieve the overall reduction required to accommodate additional functionality.

In all, 5 out of 25 software partitions were analysed. These 5 partitions are referred to below as Partitions A to E. The software for these partitions amounted to over 100,000 lines of Ada code. Three of the partitions, A, B and C were comprehensively analysed, with improvements and targets for optimization selected on the basis of the information provided by RapiTime. Optimizations were prototyped for these partitions and the RapiTime performance analysis re-run to quantify the improvements obtained. For the final two partitions, D and E, several optimizations were identified, however prototyping and further analysis awaits the next phase of the study.

The analysis process sought to achieve savings in the overall execution time schedule, in the following categories:

1. **Budget reductions:** reductions in execution time budgets and hence schedule slots made possible by more accurate analysis of partition worst-case execution times.
2. **Optimizations** at design level, sub-program level and low-level.

[2] ETUs are an arbitrary execution time unit used in this paper. The actual values are 'commercial in confidence' and are therefore not reported here.

Major savings in each of these categories are discussed in the following sections.

5.1 Budget reductions

During initial investigation of Partition A, it was found that the schedule slot (execution time budget) was significantly greater than actually required in the context of its use in the Operational Flight Program. The schedule slot had previously been increased to accommodate use of the partition in different context where it had a much longer execution time. Accurate context dependent analysis of the execution time allowed the budget to be safely reduced by 58 ETUs.

5.2 Design level optimization

Detailed analysis of Partition A revealed that over 80% of its execution time was spent copying data to a large intermediate buffer. Further investigation showed that in the context of how the software was used in the Operational Flight Program, only one response at a time was possible from any given client and thus the intermediate buffer copy was unnecessary. Removing this copy reduced the execution time of Partition A by over 62%, an overall saving of 17 ETUs.

This optimization opportunity is representative of the value of prioritising optimization opportunities. Determining that the usage of this component did not need an intermediate buffer was not obvious and required detailed discussion with various engineers responsible for the overall design of the system. This investigation would have not been performed if there had not been strong evidence of potentially large savings in execution time.

5.3 Sub-program optimization

Analysis of Partition C revealed that over 25% of the execution time of the partition was spent copying data in a loop that iterated over 2000 times. Close inspection of the code that performed this copy showed numerous redundant constraint checks. This code was replaced by a call to memcpy enabling the compiler to use more efficient code for the copy, without the large number of constraint checks. This reduced the execution time of the sub-program by over 80%, resulting in an overall reduction in the execution time of the partition of 23%, corresponding to a saving of 48 ETUs.

This optimization shows the trade-off between maintainability and code readability versus execution time. Widespread use of memcpy routines for copying data is not recommended as it makes the program less readable and less maintainable; however, in this context the change was more than justified by the significant gain in performance.

5.4 Low-level optimizations

In Partition B, RapiTime showed that a small bit-unpacking sub-program was called over 700 times on the worst-case path. Further investigation showed that the compiler generated code was not particularly efficient. Writing the Ada code in a different way allowed the compiler to produce more efficient code, reducing the execution time of the sub-program by 57%, corresponding to an overall

reduction in the execution time of Partition B of 7%, and a saving of 11 ETUs.

In general, it is not practical to do object code investigation on even a medium size program. However, using RapiTime, it is possible to identify code fragments that contribute significantly to the overall worst-case execution time. Blocks of code that are called very frequently on the worst-case path (over 700 times in this case) are often a good target for low-level optimization, as proved to be the case here.

5.4 Summary of the results

Overall, the following savings in the schedule were achieved:

- **76 ETUs** due to prototyped optimizations, including:
  - 17 ETUs from design level changes.
  - 48 ETUs from sub-program modifications.
  - 11 ETUs from low-level optimizations.
- **58 ETUs** due to identifying a reduced execution time budget for Partition A.

Total reduction in execution time **134 ETUs,** exceeding the targeted reduction of 100 ETUs.

Using RapiTime to identify candidates for optimization, it was possible to achieve reductions, amounting to **23.6%** of the execution time of the analysed partitions, whilst needing to manually examine just **1.2%** of the total lines of source code in these partitions. These 1250 lines of code were initially responsible for 29% of the overall execution time of the partitions. Design-level, sub-program-level and low-level optimizations reduced this contribution by a factor of almost 5, creating headroom for new functionality to be added without the need for expensive hardware upgrades.

	Execution Time		Improvement
Partition	**Before**	**After**	**%**
Partition A	28.2	10.6	62.4%
Partition B	140	129	7.9%
Partition C (1)	95.5	72.9	23.7%
Partition C (2)	58.1	33.2	42.9%
Total	321.8	245.7	**23.6%**

Table 1 Reduction in worst-case execution times achieved using RapiTime

The reductions in partition execution times achieved are summarised in Table 1 and illustrated in Figure 2. Partition C appears twice as it is executed twice within the major cycle of the schedule. The contexts of these two executions are however different and consequently two different context dependent execution times were derived for Partition C.

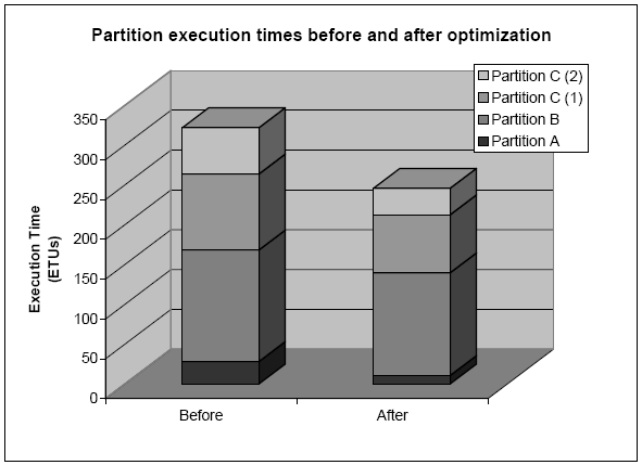

Figure 2 Reduction in worst-case execution times achieved using RapiTime

6 Summary and conclusions

During the study described in this paper, the process of using RapiTime for the Hawk AJT project was refined. A number of partitions within the Operational Flight Program of the Hawk Mission Computer were successfully analysed and significant reductions in execution time made. Overall, the improvements made put the project on track to provide the headroom necessary to incorporate additional functionality without recourse to an expensive hardware upgrade.

As part of the study, RapiTime identified that only 1.2% of the code contributed more than 29% of the overall worst-case execution time. These blocks of code were obvious targets for optimization. A detailed study of some 1250 lines of code identified specific targets for optimization and hence opportunities for execution time reduction. These optimizations were classified as: low-level, sub-program-level and design-level. The best candidates were prototyped and implemented and the new system analysed to verify the effectiveness of the changes. The optimized partitions had an execution time that was 23% smaller than before.

References

[1] BAE Systems. Hawk Jet Trainer. http://en.wikipedia.org/wiki/BAE_Hawk

[2] G. Bernat, A. Burns and M. Newby (2005), *Probabilistic Timing Analysis. An approach using Copulas*, Journal of Embedded Computing, Vol 1 no 2, pp 179-194

[3] Rapita Systems Ltd. *RapiTime White Paper*. (2005.) http://www.rapitasystems.com

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o K.U. Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Peter Dencker
c/o Parasoft Deutschland GmbH
Bayerstraße 24
D-80335 München
Germany
Phone: +49-89 4613323-15
Fax: +49-89 4613323-23
Email: dencker@parasoft.de
URL: ada-deutschland.de

Ada-France

Association Ada-France
c/o Jérôme Hugues
Département Informatique et Réseau
École Nationale Supérieure des Télécomunications
46, rue Barrault
75634 Paris Cedex 135
France
Email: bureau@ada-france.org
URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
Ada-Spain
P.O.Box 50.403
28080-Madrid
Spain
Phone: +34-942-201-394
Fax: +34-942-201-402
Email: gutierjj@unican.es
URL: www.adaspain.org

Ada in Sweden

attn. Rei Stråhle
Saab Systems
S:t Olofsgatan 9A
SE-753 21
Uppsala
Sweden
Phone: +46 73 437 7124
Fax: +46 85 808 7260
Email: Rei.Strahle@saabgroup.com
URL: www.ada-i-sverige.se

Ada in Switzerland

attn. Ahlan Marriott
White Elephant GmbH
Postfach 327
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: ada@white-elephant.ch
URL: www.ada-switzerland.org