ADA USER JOURNAL

Volume 27
Number 2
June 2006

Contents

Editorial Policy for *Ada User Journal*

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

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

Reprinted Articles

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

Editorial

In closing the June issue of the journal, memory goes back to the numerous events that the Ada community at large enjoyed this month: the annual conference in beautiful Porto, which featured an intense and interesting program, both technically and socially; the steady progress of the new language standard, some advance flavours of which the most eager users are already able to experiment with; the General Assembly of the Ada-Europe association, which saw the hand-over between a sizeable portion of the Board members: Farewell to the outgoing members, with congratulations for the accomplished services and welcome to the new members, with wishes for a productive service to the community.

This issue of the Journal features technical contributions that originate from as many as three continents: a technical article submitted by authors based in Universities at Bangalore and Salem, India, which wittingly discusses the value of design patterns in modern software engineering; a short summary of a survey that the Ada Resources Association in the USA successfully performed to gauge the size and the wealth of the Ada market; an account of the very successful Ada UK 2006 conference, which revived the tradition of an important Ada event in the UK. The rest of the issue contains the usual wealth of news and calendar events, once again edited by Santiago Urueña and Dirk Craeynest, respectively.

My best wishes for a happy reading and for the forthcoming Summer holidays.

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

News

Santiago Urueña

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

Contents

Ada-related Organizations

ARA — Technical Work on Ada 2005 Standard Completed

URL: http://www.adaic.com/news/iso-Ada05.html

Technical Group Reaches Consensus and Moves Language Amendment to Next Milestone

SALT LATE CITY, UT [May 2, 2006] Today at the Systems & Software Technology Conference, the Ada Resource Association announced the accomplishment of a major milestone in the development of the new Ada ISO standard. ISO's Ada Working Group (WG 9) has unanimously accepted the proposed amendment to the language and has forwarded it to the parent organization for an official ballot. Formal approval by ISO is expected some time later this year.

The new amendment to the language, commonly referred to as Ada 2005, culminates a collaborative international effort to enhance the 1995 version of the Ada language. The effort was sponsored in part by the Ada Resource Association, which helped support the work of the project editor, Mr. Randall Brukardt.

"Gaining WG 9 approval for the amendment to the language is a key step," said Mr. James Moore, Convener of WG 9. "The new features draw on programming language design and user experience over the past ten years, and they should serve to increase Ada's attractiveness in applications where reliability, safety, efficiency, and maintainability are demanded."

"Ada 2005 is a breakthrough in language technology," added Dr. Ben Brosgol, President of the Ada Resource Association. "It has advanced the state of the art in language design while preserving Ada's long-standing support for sound software engineering. WG 9 is to be congratulated for bringing this effort to fruition with a strong consensus on the features being added."

Ada 2005 offers significant enhancements in several areas. Improvements in the language's Object-Oriented Programming features include the addition of Java-like interfaces and traditional "object.operation" syntax. More flexible program structuring allows mutually dependent package specifications and makes it easier to interface with languages such as Java. Real-time system support includes additional task dispatching policies such as Earliest Deadline First, execution-time clocks, and handlers for task termination. The concurrency and object-oriented features are successfully unified through a new interface feature that allows implementation through either a sequential or concurrent type.

Support for safety and security is enhanced with the inclusion of the Ravenscar Profile (a tasking subset that is amenable to safety certification), syntax that avoids some common Object-Oriented Programming errors with inheritance, and a mechanism for defining language profiles. Other enhancements increase the language's general expressiveness, for example by allowing nested subprograms to be passed as run-time parameters, and by extending the predefined environment with new functionality, such as a Containers library.

About the Ada Resource Association

The Ada Resource Association (ARA) is an international Ada advocacy trade group comprising major Ada language and tool vendors. The ARA financially supports the maintenance of the Ada language standard and is committed to ensuring the continued success and expanded usage of Ada-related technology. Current ARA members are AdaCore, IBM Rational Software, Praxis High Integrity Systems, and SofCheck.

[See also "Ada 2005 Standarization Status" in AUJ 26-4 (Dec 2005), p.249. — su]

Ada-related Events

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

Mar 28 — Ada Conference UK 2006 Videos and Slides

Author: Jamie Ayre
Title: Videos from the Ada event in the UK
Date: Wednesday May 31, 2006
Source: AdaCore development log
URL: http://www.adacore.com/2006/05/31/videos-from-the-ada-event-in-the-uk/

For those of you that couldn't make it to this event, the videos of the speakers along with the slides presented are available [at http://www.adacore.com/home/ada_answers/lectures/ada_uk06 —su]

Topics covered include: Ada 2005, Ada and real-time programming, safety-critical and secure software development, among others.

[See also "March 28 — Ada Conference UK 2006" in AUJ 27-1 (Mar 2006), p.7. — su]

Nov 12–16 — SIGAda 2006

From: Leemon Baird <leemon@leemon.com>
Newsgroups: comp.lang.ada
Subject: CFP - SIGAda 2006
Date: 20 Mar 2006 12:51:05 -0800

Call for Participation — SIGAda 2006

Conference on Software Development for Safety, Security, and High Reliability Systems

November 12–16 2006, Albuquerque, NM, USA.

Submission deadline: 16 May 2006

Sponsored by ACM SIGAda

http://www.acm.org/sigada/conf/sigada2006 (Approval pending by ACM)

Constructing highly reliable software is an engineering challenge that can now be met in many domains. The SIGAda 2006 conference focuses on how the application of software engineering methods, tools and languages interrelate and on how features in Ada affect the quality of the resulting software. Papers that analyze Ada with respect to these factors or in comparison to other languages are especially welcome. SIGAda 2006 gathers industry experts, educators, software engineers, and

researchers interested in developing, analyzing, and certifying reliable, cost-effective software. Technical or theoretical papers as well as experience reports with a focus on Ada are solicited. A brief list of topics include safety and high integrity issues, real-time and embedded applications, Ada & software engineering education, Ada in other environments such as XML and .NET, Ada and other languages, metrics, standards, analysis, testing, validation, and quality assurance. For a more extensive list of topics visit the SIGAda 2006 web page.

Contributions are solicited in six categories: Technical articles, extended abstracts, experience reports, workshops, panels, and tutorials.

We openly welcome contributions from educators and students. Educator grants are available and should be applied for by 26 October 2006 (please see http://www.acm.org/sigada/conf/sigada2006/grants.html). An Outstanding Student Paper Award will be given for the best student contribution to the conference. Technical articles or experience reports from students could focus on such projects as comparing applications implemented in other languages and then re-implemented in Ada, mixed language development of applications, how Ada is used with XML or .NET applications, and/or software engineering education experiences with Ada.

Please see the full Call for Participation on the SIGAda 2006 web site for submission details. http://www.acm.org/sigada/conf/sigada2006

The deadline for submission is 16 May 2006.

Leemon Baird, SIGAda 2006 Program Chair

Ada and Education

GNAT GPL available for the GAP community

From: Jamie Ayre <ayre@adacore.com>
Date: Tue, 6 Jun 2006 17:32:37
Subject: [AdaCore] [F306-016] - GNAT GPL 2006 now available for GAP members
To: announce@adacore.com

We are pleased to announce the release of GNAT GPL 2006 available for the GAP community. To facilitate the job of distributing GNAT, we have synchronized the technology distributed to the Academic and Free Software communities. Hence, on the most popular personal platforms, your students will be able to download GNAT GPL 2006 directly from libre.adacore.com.

It is immediately available on the x86 GNU Linux, Windows, Mac OS X, SPARC Solaris, platforms. We also plan to make available on the very near future, 2 new important pieces of technology:

- a bareboard cross configuration targeted to the ERC32 along with its simulator. This is an ideal platform for introducing students to realistic embedded development on bare machines.

- a new runtime for GNU Linux platforms based on MaRTE (thanks to a cooperative effort with the Santander University team). This new runtime will be ideal for real-time courses thanks to its full support for Ada 95 Annex D. It will will also, in the future, be the base for all the new Ada 2005 annex D features.

GNAT GPL 2006 can be downloaded from the "Download" section on GNAT Tracker. Please note that, for your convenience, GNAT Tracker can now be accessed directly from AdaCore's academic page:

http://www.adacore.com/home/academia/

This new edition includes almost all of the new features introduced in the recent language revision, Ada 2005. These are described in the new Ada 2005 reference manual (now included in the GNAT GPL documentation). Highlights include:

Object-oriented features:

- abstract interfaces (AI-251)

- object operation notation (AI-252)

- nested type extensions (AI-344)

- synchronized interfaces (AI-345)

Program structure:

- unchecked union (AI-216)

- limited with clauses (AI-217)

- overriding indicators (AI-218)

- private with_clauses (AI-262)

- aggregates for limited types (AI-287)

- partial parametrization of formal packages (AI-317)

- limited and anonymous access return types and the extended return statement (AI-318)

- null procedures (AI-348)

Libraries:

- directory operations (AI-248)

- container library (AI -302)

- time operations (AI-351)

- environment variables (AI-370)

Concurrency:

- the Ravenscar profile (AI-249)

- timing events (AI-297)

- priority-specific dispatching (AI-355)

Enhanced access types:

- generalized used of anonymous access types (AI-230)

- anonymous access to subprogram types (AI-254)

- current instance rule for access types (AI-382)

SPARK Training

URL: http://www.praxis-his.com/sparkada/training.asp

Public Course Dates for 2006 — UK

Course 1 – "Software Engineering with SPARK"

11th – 14th September 2006 at the Praxis Offices in Bath.

Course 2 – "Black-Belt SPARK"

19th – 21st September 2006 at the Praxis Offices in Bath.

Course 3 – "High-Integrity Concurrent Software Design with RavenSPARK"

15th September 2006 at the Praxis Offices in Bath.

Course 4 – "UML to SPARK"

15th September 2006 at the Praxis Offices in Bath.

[See also same topic in AUJ 26-4 (Dec 2005), p.232. — su]

Public Ada 95 Courses

From: Ed <colbert@abssw.com>
Date: 19 May 2006 12:47:26
Subject: [Announcing] Public Ada 95 Courses 12-16 June in Carlsbad CA
Newsgroups: comp.lang.ada

Absolute Software will be holding a public Ada 95 course during the week of 12 June 2006 in Carlsbad, CA. You can find a full description and registration form on our web-site, www.abssw.com. Click the Public Courses button in the left margin. (We also offer courses on software architecture-based development, safety-critical development, object-oriented methods, and other object-oriented languages.)

If there is anything you'd like to discuss, please call, write, or send me E-mail.

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

Ada-related Resources

PragmAda's New Home

From: PragmAda Software Engineering <pragmada@mchsi.com>
Date: Mon, 27 Mar 2006 19:06:11
Subject: PragmAda's New Home
Newsgroups: comp.lang.ada

PragmAda Software Engineering has a new web address:

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

The PragmAda Reusable Components are at:

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

and the Mine Detector game at:

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

You may send e-mail to PragmAda at the "From" address of this message.

Jeffrey R. Carter, President, PragmAda Software Engineering

[See also "PragmARC — PragmAda Reusable Components" in this issue — su]

Ada-related Tools

GNAT GPL 2006 Edition

From: Jamie Ayre <ayre@adacore.com>
Date: 14-jun-2006 12:07
Subject: [AdaCore] GNAT GPL 2006 now available
To: announce@adacore.com

We are pleased to announce the release of GNAT GPL 2006. This new edition includes almost all of the new features introduced in the recent language revision, Ada 2005. These are described in the new Ada 2005 reference manual (now included in the GNAT GPL documentation). Highlights include:

Object-oriented features:

- abstract interfaces (AI-251),

- object operation notation (AI-252)

- nested type extensions (AI-344)

- synchronized interfaces (AI-345)

Program structure:

- unchecked union (AI-216)

- limited with clauses (AI-217)

- overriding indicators (AI-218)

- private with_clauses (AI-262)

- aggregates for limited types (AI-287)

- partial parametrization of formal packages (AI-317)

- limited and anonymous access return types and the extended return statement (AI-318)

- null procedures (AI-348)

Libraries:

- directory operations (AI-248)

- container library (AI -302)

- time operations (AI-351)

- environment variables (AI-370)

Concurrency:

- the Ravenscar profile (AI-249)

- timing events (AI-297)

- priority-specific dispatching (AI-355)

Enhanced access types:

- generalized used of anonymous access types (AI-230)

- anonymous access to subprogram types (AI-254)

- current instance rule for access types (AI-382)

GNAT GPL 2006 comes with the latest versions of the GNAT IDE, GPS 3.1.3 and PolyORB. For more information on these technologies please visit http://www.adacore.com/home/gnatpro/updates

It is available on the Linux, Windows, and Mac OS X platforms.

GNAT GPL 2006 can be downloaded from the "Download GNAT GPL Edition" section on libre.adacore.com

[See also "GNAT GPL 2005 Edition" in AUJ 26-3 (Sep 2005), pp.153–154. — su]

The GNU Ada Compiler

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Thu, 13 Apr 2006 20:16:45
Subject: [gnuada] Help needed for MS-Windows version
Newsgroups: comp.lang.ada

The GNU Ada project needs some help from experienced Ada MinGW users. Although we have tried to produce a working MS-Windows version, it did not work out.

So here is our call for help. If you are interested you can start by looking on how far we came:

http://gnuada.sourceforge.net/pmwiki.php/Install/MS-Windows

Join our discussions on:

http://sourceforge.net/forum/forum.php?forum_id=40858

http://sourceforge.net/forum/forum.php?forum_id=40860

Or get right down to it by downloading the current make scripts:

http://sourceforge.net/svn/?group_id=12974

and try them out.

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Thu, 13 Apr 2006 20:02:29
Subject: [gnuada] AIX available.
Newsgroups: comp.lang.ada

Thank's to Karl Nyberg we have one more operating system on our portfolio: IBM AIX 5.1 for PowerPC. That brings us to 10 OSs — Thanks Karl.

From: Björn Persson <rombo.bjorn.persson@sverige.nu>
Date: Sat, 08 Apr 2006 23:09:13
Subject: Re: [gnuada] R4 update available.
Newsgroups: comp.lang.ada

And now there are Fedora versions of the RPM packages, for both Fedora 4 and Fedora 5 on i386.

A warning though: There is a bug in the latest kernel updates for both Fedoras. Gnat doesn't work at all on these kernels. Until this bug has been fixed you have to stick to the Linux 2.6.15 packages when using Gnat.

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Thu, 30 Mar 2006 20:34:43
Subject: [gnuada] R4 update available.
Newsgroups: comp.lang.ada

We have updated the GNAT distributions. It is a minor fix mainly some sources where missing in the devel packages for GPS. Those have been added. Sadly we where not able to compile GPS after all — neither with GNAT/GCC nor with GNAT/GPL. I find it a bit disturbing that GPS can not be compiled with any GNAT I can lay my hands on.

But AWS is included for platforms where compiles where possible. See:

http://gnuada.sourceforge.net/pmwiki.php/Main/3rdPartyBugs

We also have a new version for MS-Windows cygwin for you to try. Since cygwin is closer to Linux than mingw so it was easier to create. We could really do with some dedicated Windows packager.

For installation see:

http://gnuada.sourceforge.net/pmwiki.php/Install/Cygwin

It's a Wiki — share your experience. But do not use the names of well known pharmaceutical products. They are on the list of banned words ;-).

AVR-Ada

From: Rolf <rolf.ebert@gmx.net>
Date: 15 May 2006 00:18:48
Subject: [Announce] AVR-Ada V0.4.0 released
Newsgroups: comp.lang.ada

We are proud to announce a new release of AVR-Ada, one of the first GCC based Ada compilers targeting 8-bit microcontrollers.

You get the project description and some documentation at:

http://avr-ada.sourceforge.net/

The Sourceforge development pages with the download section are at:

http://www.sourceforge.net/projects/avr-ada/

AVR-Ada is available in source and binary form. Binary packages of the cross compiler hosted on Linux and Windows are available in the download area. A future release of WinAVR (winavr.sourceforge.net) will probably also include AVR-Ada.

Feel free to join the mailing list at:

http://lists.sourceforge.net/mailman/listinfo/avr-ada-devel

It has quite low traffic.

Please use SF's bug reporting and feature request system for guiding future development of AVR-Ada.

Status

The goal of the AVR-Ada project is to make the gcc based Ada compiler GNAT available for the AVR microcontrollers.

More specifically the project provides

- a GNAT compiler based on the existing AVR and Ada support in gcc

- a minimalistic Ada runtime system

- a useful AVR specific support library

The current distribution of AVR-Ada is V0.4.0. It is based on gcc-3.4.6 and gcc-4.1.1 (prerelease). In the AVR-Ada project we rarely have problems with the Ada compiler itself. It is quite stable.

The Ada run time system (RTS) on the other hand is for the most part not even a *run* time system. It is more a compile time system :-). Most files in the RTS are only needed at compile time. As a consequence we don't have support for exceptions nor for tasking (multithreading).

There is some AVR specific support. Type and interface definitions, timing routines, eeprom access, UART, and most importantly the necessary definitions for most AVR parts.

Some sample programs in the apps/ directory show how to use the compiler and the library. This includes the demo programs from the avr-libc distribution and some of Peter Fleury's example programs (http://homepage.sunrise.ch/mysunrise/peterfleury/avr-software.html) translated to Ada.

The documentation is still low and consists only of the pages at avr-ada.sourceforge.net. A copy of the pages is in the directory AVR-Ada-0.4.0/web/ for offline reading. Feel free to ask any question on the mailing list.

News

V 0.4.0 (2006-05-11)

The part description Ada specs are now based on Atmel's XML part description files of AVR Studio 4.12 build 473 SP 2 + Atmega644p.

Limited support for new devices: at90can32 at90can64 at90pwm2 at90pwm3

at90usb1287 atmega1280 atmega1281 atmega164p atmega165 atmega165p

atmega169p atmega324p atmega325 atmega3250 atmega329 atmega3290

atmega406 atmega640 atmega644 atmega644p atmega645 atmega6450

atmega649 atmega6490 attiny24 attiny25 attiny261 attiny44 attiny45

attiny461 attiny84 attiny85 attiny861

We build part specific runtime systems (RTS) now. Old Makefiles have to be adjusted!

The AVR-lib has new packages: Watchdog, Sleep, Int_Img. (Note that not all packages have been ported to all devices).

Updated scripts to build AVR-Ada with gcc-3.4.6 and gcc-4.1.0 are now located in tools/build/.

New script (wizard) to generate a ready-to-compile project directory with all necessary files (tools/mk_ada_app/).

New examples (largedemo, debounce) are located in apps/.

New bug fixes and workaounds for gcc-3.4 and gcc-4.1 are located in patches/.

[See also same topic in AUJ 26-1 (Mar 2005), p.10. — su]

Updates for Fuzzy sets for Ada, and Simple components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 25 May 2006 11:50:48
Subject: ANN: Cumulative update: Fuzzy sets, Measurements units, Components
Newsgroups: comp.lang.ada

Cumulative update:

Fuzzy sets for Ada 4.2

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

Units of measurement for Ada v2.1

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

Simple components 2.2

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

Things are now compilable with GNAT 2005, GCC 4.0.2 (20051125), some minor bug fixes and extensions made.

[See also "Fuzzy sets for Ada" in AUJ 26-4 (Dec 2005) p.237 and "Simple components" in AUJ 26-3 (Sep 2005) p.152. — su]

GLOBE_3D — 3D Engine

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Wed, 29 Mar 2006 00:04:17
Subject: Ann: GLOBE_3D, now running under Linux (Upload: 25-Mar-2006)
Newsgroups: comp.lang.ada

GLOBE_3D means "GL Object Based Engine for 3D".

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

New: runs under Linux, thanks to Marc Criley (GL/GLU/GLUT bindings with appropriate Import pragma for Linux)

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

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Mon, 15 May 2006 21:18:55
Subject: Ann: GLOBE_3D, (Upload: 15-May-2006)
Newsgroups: comp.lang.ada

News:

- full-screen mode: now the mouse functions as expected in a game: pointer invisible, no limitation against screen borders

- tools/max2ada.ms, the export script from GMax / 3D Studio Max was improved a lot, especially tiled textures are mapped exactly

- random extruded surface generator — e.g., you can generate a Sci-Fi city with a minimal effort.

One single source set — without any conditional compilation — for all platforms and compilers. Tested on Windows and Linux.

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Tue, 16 May 2006 21:41:29
Subject: Re: Ann: GLOBE_3D (v. 14 mai 2006)
Newsgroups: fr.comp.lang.ada

[Translated from French – su]

> Weird.They talk about compatibility with 3D scenes max and not about Blender. Did I get anything wrong?

If you kind of imply that that's incompatible with Blender, then yes you got it wrong: that 3D engine is compatible with all modeling tools around, but the code to export scenes still is to be written. To that end there is just a single solution: roll up your sleeves!

As far as Blender is concerned you'll certainly never fall short of sources of inspiration …

http://www.google.com/search?&q=export+scene+blender

That said, if the scripting language for Blender ever gets to be as hopeless as that in 3DS, then it won't be such fun anymore!

Much the same as exporting scenes in VRML or 3DS as intermediate formats, you can pass them on into wrl2ada and max2ada.

OpenALada

From: Aurele <aurele.vitali@gmail.com>
Date: 15 May 2006 19:07:32
Subject: OpenALada v1.4
Newsgroups: comp.lang.ada

The OpenALada binding has been updated for OpenAL v1.1. Visit http://www.openalada.com for details.

[See also same topic in AUJ 26-2 (Jun 2005), p.75. — su]

YAML data serialization format parser

From: Y.Tomino <demoonlit@panathenaia.halfmoon.jp>
Date: Thu, 11 May 2006 04:33:33
Subject: Re: YAML
Newsgroups: comp.lang.ada

> A quick google for YAML and Ada turned up nothing. Does anyone know of tools for using YAML in Ada?

I've made a reader/writer subset of YAML for myself.

http://panathenaia.halfmoon.jp/alang/dyayaml.7z

It needs http://panathenaia.halfmoon.jp/alang/ase.7z (my personal packages)

I did not write any documentation, sorry.

ASnip — Ada source code decorator

From: Georg Bauhaus <bauhaus@futureapps.de>
Subject: ANN: Ada source code decorator
Date: Tue, 23 May 2006 14:59:29
Newsgroups: comp.lang.ada

ASnip reads snippets of Ada source text, correct or incorrect, and produces output suitable for printing, viewing, or including in web pages (also in the Ada Wikibook).

* HTML - use your own style sheets, or the ones provided

* Text - for regression testing, really

* TeX - via Knuth's WEB macros (Try lgrind with LaTeX)

* WiKiBook - adds markup used for the Ada Wikibook, automatically links tokens to the Ada reference

(* RTF - time permitting)

(* XML - time permitting, using Simon Wright's ASIS based GIs.)

http://home.arcor.de/bauhaus/Tools/ASnip/

Source code to XML

From: Marc A. Criley <mc@mckae.com>
Date: Tue, 23 May 2006 15:40:15
Subject: Re: Can I get access to an AST of parsed Ada code?
Newsgroups: comp.lang.ada

> I have legacy Ada and C++ code, and I need to export this code into an XML format so I can create a tool to analyze and work with it. I tried various tree-dump-* commands with g++ and didn't find anything that looked to be readable or anything that was at a high level like my code (GIMPLE was much to low level for me because I want to maintain variable names and such).
>
> Is there a way I can access the parser for either C++ or Ada so that I can access a high level parsed version of my code? If not, can a parser dump command be recommended?

For C++ you can use GCC_XML from http://www.gccxml.org/, and for Ada you have ASIS2XML, http://www.pushface.org/asis2xml.

For Ada the code will have to be compilable by a version of GNAT with its corresponding ASIS implementation.

From: Ira D. Baxter <idbaxter@semdesigns.com>
Date: 24 May 2006 06:56:16
Subject: Re: Can I get access to an AST of parsed Ada code?
Newsgroups: comp.lang.ada

My understanding is that GCC_XML does not output function bodies (see their web page).

The DMS Software Reengineering Toolkit has robust parsers for many dialects of C++, including ANSI, GNU, and MS Visual Studio 2005, and parsers for Ada 83 and 95. DMS has an option to dump the XML produced by the parsers, and so would satisfy the OP's request.

See: http://www.semanticdesigns.com/Products/DMS/DMSToolkit.html

From: Marc A. Criley <mc@mckae.com>
Date: Wed, 24 May 2006 12:16:48
Subject: Re: Can I get access to an AST of parsed Ada code?
Newsgroups: comp.lang.ada

Yep, that's correct. So it depends on what one needs to do with the C++. If the need is to work mostly with declarations (which is all I care about), then GCC_XML should suffice. If the analysis needs to be more comprehensive, processing the executable content as well, then one has to look elsewhere.

[See also "ASIS2XML" in AUJ 25-4 (Dec 2004), p.191. — su]

Forth interpreter in Ada

From: Samuel Tardieu <sam@rfc1149.net>
Date: 30 May 2006 12:38:46
Subject: Forth embeddable interpreter written in Ada
Newsgroups: comp.lang.ada

For the French robotics cup, I wrote a Forth interpreter in Ada which you can easily embed in your application. Look at http://tinyurl.com/ru4qv to get it under the GPLv2. No doc at all, it was just a dirty hack to be able to test the robot functionalities, I will add some docs later if time permits.

The interpreter was tested on Linux/x86 and Linux/sh4 (which we ran on the two robot boards).

PS/ Before you ask, our ranking was near the middle, much better than last year and probably much worse than next year. The Ravenscar profile allowed us to get very clean, efficient and well structured code.

Ada-related Products

AdaCore — GNAT Pro 5.04a

From: Jamie Ayre <ayre@adacore.com>
Date: Tue, 07 Mar 2006 18:23:32
Subject: [AdaCore] F213-023 - Announcing immediate availability of GNAT Pro 5.04a - batch 2
To: announce@adacore.com

AdaCore are pleased to announce the immediate release of 5.04a for the following platforms:

ppc-vxw-solaris

ppc-vxw-windows

ppc-elf-solaris

This release comes with a corrective version of the GNAT Programming Studio for all native platforms on which it is supported.

This release also includes PolyORB 2.0 for the following platforms:

sparc-solaris

x86-linux

pa-hpux

The following UNIX packages have been repackaged to address the warning on installation issue reported with 5.04a release that was distributed earlier this year:

alpha-tru64

ia64-hpux

mips-irix

pa-hpux-11

ppc-aix-5.1

sparc-solaris

x86-solaris

The distributions can be downloaded as usual using GNAT Tracker. Note that, for your convenience, GNAT Tracker can now be accessed directly from Adacore's home page (http://www.adacore.com). You may also want to take a moment to discover our new web site and in particular the Developer Center.

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

[See also "AdaCore — GNAT Pro 5.04" in AUJ 27-1 (Mar 2006) p.11–12 and "AdaCore — GNAT Pro 5.03a" in AUJ 26-1 (Mar 2005) pp.13–14. — su]

AdaCore — GNAT Pro on x86-64

http://www.adacore.com/2006/04/18/adacore%e2%80%99s-gnat-pro-brings-ada-to-x86-64-gnulinux/

Tuesday April 18, 2006

AdaCore's GNAT Pro Brings Ada to x86-64 GNU/Linux

GNAT Pro and Ada ease customer transition to high-performance, 64-bit applications.

AdaCore today announced the availability of its flagship GNAT Pro Ada development environment on the x86-64 platform. The product is available on two primary GNU/Linux operating systems — Red Hat® Enterprise Linux® v. 4, and SUSE® Linux Enterprise Server 9 — on the Intel EM64T and the AMD64 processors. AdaCore's porting of GNAT Pro to 64-bit platforms reflects the market's natural progression from 32-bit to 64-bit computing. Because 64-bit architectures overcome the 4 GB memory space limitations of standard 32-bit platforms they are able to handle larger and more demanding applications. In addition, recent x86 architecture improvements, including advanced multi-core processing, now provide exceptional processing power, multi-threaded throughput, and better performance. As a result, the x86-64 platform is quickly gaining mainstream popularity with vendors of compute-intensive, memory-hungry applications, such as servers, databases, and telecommunications. And by also providing support for 32-bit execution, the processor eases the transition to 64-bit computing.

"We have received requests from major enterprise players to support Ada on x86-64 configurations, and we anticipate continued industry demand in the future," said Robert Dewar, CEO of AdaCore. "With this port of GNAT Pro to x86-64, our customers will be able to break free from the limitations of 32-bit platforms, while benefiting from the full complement of AdaCore tools and services they currently enjoy with other GNAT Pro supported platforms."

About GNAT Pro

GNAT Pro is a robust and flexible open-source Ada development environment based on the GNU GCC compiler technology. It comprises a full Ada compiler, an Integrated Development Environment (GPS, the GNAT Programming Studio), a comprehensive toolset including a visual debugger, and a useful collection of libraries / bindings. GNAT Pro allows development of pure Ada applications as well as Ada components in multi-language systems. It is distributed with complete source code, and is backed by rapid and expert support service. GPS is available on a wide range of host environments for both native and cross-development using GNAT Pro, including UNIX, Windows and GNU/Linux.

GNAT Pro supports the major new features in the Ada 2005 revision of the Ada programming language. With over 120 enhancements over the previous release of the technology, GNAT Pro is the best choice for reliable and efficient software, across a wide spectrum of applications, including high-integrity systems.

About AdaCore

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

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

AdaCore — GNAT Pro for HP OpenVMS

http://www.adacore.com/2006/05/01/gnat-pro-now-available-for-hp-openvms-on-hp-integrity-servers/

Monday May 1, 2006

GNAT Pro Now Available for HP OpenVMS on HP Integrity Servers

GNAT Pro helps customers meet stringent requirements for mission-critical software systems

SALT LAKE CITY, USA — Today at the Systems & Software Technology Conference AdaCore announced the immediate availability of its flagship GNAT Pro Ada development environment for HP OpenVMS on HP Integrity servers. GNAT Pro for OpenVMS on HP Integrity servers is tailored to the needs of developers who require reliability, performance and maintainability in their software. It can be used for a broad spectrum of applications, including database systems, device control and transportation.

GNAT Pro for OpenVMS on HP Integrity servers comprises a full Ada compiler, a comprehensive toolset, and supplemental libraries and bindings. It allows developers to build pure Ada applications as well as Ada components in multi-language systems. The product not only implements Ada 95, but it also supports the Ada 83 subset and a large set of the new Ada 2005 features. It is optimized to take full advantage of the performance and scalability of the 64-bit HP Integrity server architecture, allowing full use of the large address space.

AdaCore has been providing Ada products on OpenVMS platforms since 1998, when the company introduced its GNAT Pro environment for HP AlphaServer systems.

"GNAT Pro and OpenVMS have always been a natural mix because of their shared emphasis on reliability and robustness," said Robert Dewar, CEO of AdaCore. "With GNAT Pro now available for HP Integrity servers running OpenVMS, developers on the platform have the opportunity to use Ada, the language best suited for mission-critical systems."

GNAT Pro is especially useful for systems comprising many thousands of modules and millions of lines of code. Its robust system architecture allows scalability based on program size and does not degrade abruptly when a fixed capacity is reached. Its project facility provides a flexible framework for organizing large, multi-person development efforts.

"We are delighted that AdaCore has chosen to strengthen its commitment to HP OpenVMS and extend the value of HP Integrity servers to its customers," said Ann McQuaid, general manager of the OpenVMS Group, Business Critical Servers, HP. "The deployment of Ada products on HP Integrity servers enables customers to experience the synergy between the mission-critical enterprise computing strengths of OpenVMS and the secured availability of the HP Integrity platform."

GNAT Pro is highly compatible with the HP Alpha-hosted Ada 83 compiler (formerly known as "DEC Ada"), implementing HP-specific pragmas and attributes, following HP Ada's representational conventions, and providing a binding to the HP Ada predefined library. GNAT Pro tool invocation has the OpenVMS "look and feel", using standard OpenVMS syntax and conventions.

GNAT Pro for OpenVMS implements both Ada-specific and platform-dependent optimizations, and the run-time library has been designed to map Ada's dynamic features directly and efficiently onto the

underlying OpenVMS services. The product features a comprehensive tool suite, including a pretty-printer, program browser and program metrics generator.

AdaCore distributes GNAT Pro with complete source code, and backs the product with rapid and expert support service.

Availability

GNAT Pro for OpenVMS is currently available for HP Integrity servers. Pricing for GNAT Pro subscriptions starts at $14,000. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations.

AdaCore — GNAT Pro for VxWorks Simulator

URL: http://www.adacore.com/2006/05/16/vx-sim-release/

Tuesday May 16, 2006

GNAT Pro Supports Simulator for VxWorks 6 and VxWorks 653

ORLANDO, FL, USA — Today at the Wind River Worldwide User Conference AdaCore announced that its flagship GNAT Pro Ada development environment now supports the latest versions of the Wind River® VxWorks Simulator, a prototyping and simulation tool for VxWorks® 6 and VxWorks 653 applications in the Wind River" Workbench development suite. VxWorks Simulator (formerly known as VxSIMTM) enables application development and testing without the need for target hardware – either before hardware is available, or to reduce the number of targets required, thereby lessening development cost. It is fully integrated into the Wind River Workbench development suite for execution of VxWorks applications, allowing complete configuration, execution and debugging control through standard interfaces on the host platform.

AdaCore's GNAT Pro is well established on Wind River platforms, with several hundred customers already using GNAT Pro for VxWorks. GNAT Pro for VxWorks 6 is currently targeted to the Wind River" General Purpose Platform, VxWorks Edition, on the PowerPC, from Windows, GNU/Linux and Solaris host environments. GNAT Pro for VxWorks 653 is targeted to the Wind River" Platform for Safety Critical ARINC 653 supporting Windows and Solaris host environments. AdaCore also offers plug-in support for Wind River Workbench. Workbench is an Eclipse-based development suite optimized for device software and supported by Wind River's worldwide professional services and support organization.

"We welcome AdaCore's expanded support for Wind River's Workbench development suite and simulation products," said Chip Downing, industry marketing manager for Aerospace and Defense at Wind River Systems. "These integrated products continue to expand the design and debug choices for our large and growing customer base. Combined with our expert worldwide support organization, our companies lead the industry with robust multi-language development platforms for high-reliability systems."

"Support for the VxWorks Simulator is a natural addition to AdaCore's GNAT Pro offerings for the Wind River Workbench development suite and VxWorks-based platforms," said Robert Dewar, CEO of AdaCore. "It also demonstrates the commitment we bring to our partnership with Wind River by continually delivering new tools and support services to complement their real-time device software environments. In addition, our joint customers can continue to enjoy an ever-increasing selection of robust and proven design, debug and deployment solutions."

GNAT Pro allows users the flexibility to choose between two powerful host development environments. The first is GNATbench, a GNAT Pro plug-in developed in close collaboration with Wind River specifically for the Wind River Workbench development suite to facilitate multi-language development, sophisticated editing, browsing, debugging, comprehensive compilation, as well as prototyping and simulation for advanced VxWorks systems creation. The second is GNAT Programming Studio (GPS), a sophisticated IDE that is seamlessly integrated with the Wind River VxWorks real-time operating system (RTOS) and provides a one-click switch between native and cross environments.

About GNAT Pro for VxWorks

GNAT Pro for VxWorks includes implementation of all versions of Ada: Ada 2005, Ada 95, Ada 83; mixed-language support, allowing composition of applications comprising Ada, C, and C++; full source for GNAT Pro, allowing users to see how the run-time libraries implement dynamic Ada features in the context of VxWorks, whether in kernel or user mode; Ada run-time features (memory management, tasking, I/O) that map directly and efficiently onto the underlying VxWorks routines; an extensive GNAT library; and an Ada unit testing framework (Aunit).

GNAT Pro open standard GCC compiler technology offers backward compatibility with VxWorks 5.x to ease migration for customers choosing to transition to VxWorks 6 at their own pace.

Availability

GNAT Pro for VxWorks 6 and GNAT Pro for VxWorks 653 are currently available and targeted to the Wind River General Purpose Platform, VxWorks Edition, on the PowerPC, from Windows, GNU/Linux and Solaris host environments, and to the Wind River Platform for Safety Critical ARINC 653 from the Windows and Solaris host environments respectively. Implementations for other targets are in progress. Please check the AdaCore website or contact a sales representative for news on availability, specific configurations, or further details.

[See also "AdaCore — Support for VxWorks 6" in AUJ 26-4 (Dec 2004), pp.240–241. — su]

AdaCore — Ada 2005 Preview release

From: Jamie Ayre <ayre@adacore.com>
Date: Tue, 30 May 2006 10:35:34
Subject: [AdaCore] [F502-014] Preview release for new Ada 2005 features
To: announce@adacore.com

The implementation of Ada 2005 in GNAT is almost complete. We are pleased to announce the availability of a preview of the next release allowing beta testing of those new features as described by the new Ada 2005 reference manual (now included in the GNAT Pro documentation). Highlights include:

Object-oriented features: abstract interfaces (AI-251), object operation notation (AI-252), nested type extensions (AI-344) synchronized interfaces (AI-345)

Program structure: unchecked union (AI-216), limited with clauses (AI-217), overriding indicators (AI-218), private with_clauses (AI-262), aggregates for limited types (AI-287), partial parametrization of formal packages (AI-317), limited and and anonymous access return types and the extended return statement (AI-318), null procedures (AI-348)

Libraries: directory operations (AI-248), container library (AI -302), time operations (AI-351), environment variables (AI-370)

Concurrency: the Ravenscar profile (AI-249), timing events (AI-297), priority-specific dispatching (AI-355)

Enhanced access types: generalized used of anonymous access types (AI-230), anonymous access to subprogram types (AI-254), current instance rule for access types (AI-382)

The full list can be found on GNAT Tracker in the "Ada 2005 implemented in GNAT Pro" tab of the "Release notes" section.

The implementation of some of these features is ongoing (e.g. the extended

return statement) or depends on specialized OS facilities (e.g. priority-specific dispatching). Nevertheless, we encourage our users to experiment with the new language, which reflects a decade of experience with Ada 95, and adds substantial expressive power and increased safety to it.

In order to participate in this Beta Program, please download the Ada 2005 GNAT Pro preview using GNAT Tracker (select the version "5.05w Ada 2005 preview") and send your questions or issues using the regular GNAT Pro report mechanism. It is available on the Linux, Solaris, and Windows platforms.

Note that, for your convenience, GNAT Tracker can now be accessed directly from Adacore's home page (http://www.adacore.com).

Aonix — ObjectAda Update

From: Owner-Intel-ObjectAda <owner-intel-objectada@aonix.com>
Date: Mon, 20 Mar 2006 17:26:52
To: intel-objectada@aonix.com
Subject: Intel-OA: New ObjectAda 7.2.2 Update

A new update for Aonix ObjectAda for Windows 7.2.2, 1102V722-U21, is now available at http://www.aonix.com/ada_patches.html.

Please see the Release Notes for further details on the corrections made and installation instructions. The release notes can be viewed at ftp://ftp.aonix.com/pub/adats/outgoing/1102/7.2.2/U21/1102V722-U21.Release_Notes.

Downloading ObjectAda updates requires a password which can be obtained from your local Aonix Customer Support department. Please note that a current maintenance agreement is required to obtain the password.

For information on obtaining or renewing a maintenance agreement, please contact your nearest Aonix Sales office. For contact information see:

http://www.aonix.com/contact_us.html

[See also "Aonix — ObjectAda 8.2 for Windows" in AUJ 26-4 (Dec 2005), p.242. — su]

DDC-I — SCORE for RTX

http://www.ddci.com/display_news_item.php?filename=news_first_ada_rtx_release.php

DDC-I Announces Industry's First Ada Environment For RTX-based Windows Real-Time Systems

Mixed Ada, C and Embedded C++ applications can now run in real time on Windows systems

Phoenix, AZ. May 1, 2006. DDC-I, a leading supplier of development tools for safety-critical applications, today announced the availability of its SCORE® Integrated Development Environment (IDE) for Ardence's RTX, a real-time enabling technology for Windows applications. SCORE® is fully integrated with RTX version 6.x, a real-time extension to Windows that provides deterministic real-time multitasking, interrupt handling, and other real-time features. Now, Ada and mixed Ada/C/Embedded C++ applications developed using the SCORE® IDE can run in real time on Windows systems equipped with RTX.

"RTX transforms Windows systems into real-time platforms suitable for a wide range of applications, including mil/aero, industrial control, and telecom," said Bob Morris, president and CEO of DDC-I. "SCORE® is the first IDE for mixed Ada, C, and Embedded C++ development that lets designers take full advantage of these real-time Windows platforms."

"Combining the SCORE IDE with Ardence RTX provides an excellent solution for building deterministic control into embedded systems," said Stephen Woodard, Ardence senior vice president of global operations. "The SCORE IDE helps our customers simplify development and add system functionality, which is what makes DDC-I an important technology partner for Ardence."

SCORE® is a multi-language, object-oriented IDE for developing and deploying safety-critical applications. SCORE provides optimizing compilers for Ada, C, Embedded C ++, and Fortran77, all of which pass the applicable ACATS, PlumHall, Perennial, and FCVS compiler validation suites.

The SCORE® IDE features an intuitive GUI with a color-coded source editor, project management support, and automated build/make utilities. SCORE's multi-language, multi-window, symbolic debugger recognizes C/EC++, Ada and Fortran syntax and expressions, and can view objects, expressions, call chains, execution traces, interspersed machine code, machine registers, program stacks, etc. The debugger supports full Ada-level debugging, including constraints, attributes, tasking, exceptions, and break on exceptions and tasking events. The debugger is non intrusive, can debug at the source or machine level, and can be enabled without changing the generated code.

SCORE® supports full debugging of RTX applications running in both the Win32 and RTSS (real-time subsystem) environment. The debugger can start processes on the local computer if it is running RTX, or communicate with a remote computer running Windows XP (or Windows XP Embedded) and the RTX environment. SCORE® supplies a debug agent that runs in the RTSS environment, and a communication layer utility that bridges the Win32 and RTSS environments. These provide all of the functionality that the debugger needs to support local/remote debugging of RTX applications.

SCORE® supports a bare run-time system certifiable to DO-178B, and an enhanced bare run-time system for use in a simulated or emulated environment. The SCORE run-time can also be linked with popular real-time operating systems (RTOSes) and native operating systems like Windows. In the SCORE/RTX integration, which provides full Ada support, DDC-I has mapped the SCORE Ada run time to RTX. Here, Ada tasks become RTX threads, and the run-time system targets the RTX API instead of the Win32 API.

RTX is a high-performance extension to the Windows operating system that enables Windows applications to run in real time. Occupying just 250 kbytes of RAM, RTX supports flexible round-robin and pre-emptive scheduling (with priority inversion avoidance), and provides precise control over IRQs, I/O and memory resources, ensuring that specified time-critical tasks execute with proper priority and 100% reliability. RTX also features a WinSock compliant TCP/IP stack that is independent of Windows, and a high-speed interprocess communications (IPC) mechanism with no limitation on data message size.

RTX operates at Windows Ring 0, providing real-time services that enable Windows applications to process sustained interrupt rates of up to 30 kHz with an average IST latency of less than one microsecond. RTX is a true Windows extension, utilizing all the standard Windows conventions, including APIs, memory management, SRIs, mutexes, and semaphores that are familiar to Windows developers. RTX applications can take full advantage of the memory protection mechanisms offered by Windows and the Intel architecture in Ring 3. Once developers complete the debug process and ensure that memory pointers and arrays are valid, the RTX application can be recompiled to run in Ring 0, where it can leverage RTX's real-time services.

SCORE for RTX is available immediately. Pricing starts at $5000 for a single developer's seat. About DDC-I, Inc. DDC-I, Inc. is a global supplier of software development tools, custom software development services, and legacy software system modernization solutions, with a primary focus on safety-critical applications. DDC-I's customer base is an impressive "who's who" in the commercial, military, aerospace, and safety-critical industries. DDC-I offers compilers, integrated development environments and run-time systems for C, Embedded C++, Ada, JOVIAL and

Fortran application development. For more information regarding DDC-I products, contact DDC-I at 1825 E. Northern Ave., Suite #125, Phoenix, Arizona 85020; phone (602) 275-7172; fax (602) 252-6054; e-mail sales@ddci.com or visit www.ddci.com.

Praxis HIS — SPARK Toolset 7.31

URL: http://www.praxis-his.com/sparkada/release7p3.asp

SPARK Release 7.31

April 2006

Praxis High Integrity Systems is pleased to announce the immediate availability of Release 7.31 of the SPARK language and the SPARK toolset.

Release 7.31 includes many significant improvements, including:

VC Generation improvements in the presence of semantic and data-flow errors.

Support for full-range of IEEE 64-bit floating point values in the configuration file.

A new Examiner switch that produces explanations of errors and warnings on-screen and in the listing files.

Better error messages for common syntax errors.

Relaxation of the rule requiring qualification of modular literals.

Support for proof rules involving the 'Size attribute.

Correct order or declaration in FDL files for type-announced and private types.

Support for the use of pragma Import to complete an external own variable.

Significant new Simplifier tactics for modular and rational inequalities.

Support for user-defined proof rules for the Simplifier.

Port of the Simplifier and Checker to the SICSTUS PROLOG compiler. Both are significantly faster as a result.

Full details of all language and tool changes can be found in the release notes for releases 7.3 and 7.31.

[See also "Praxis HIS — SPARK Toolset 7.3" in AUJ 27-1 (Mar 2006), pp.15–16. — su]

Vector Software — VectorCAST 4.0

URL: http://www.vectors.com/pdf/vector_40.pdf

Vector Software announces version 4.0 release.

North Kingstown, RI – May 15, 2006 – Vector Software, the leading provider of software test tools for embedded systems, today announced the release of VectorCAST version 4.0

Version 4.0 highlights

• Integration Testing — VectorCAST 4.0 supports integration testing. This means that you can effectively test an entire sub-system, or application, using the same techniques that you use for unit testing. All existing tool functionality is supported for integration testing, including point-andclick test case editing, automated regression testing, and code coverage analysis.

• Common GUI for VectorCAST/C, VectorCAST/Ada and VectorCAST/Cover — It is now possible to create, or open, any type of VectorCAST project from the same GUI, this allows you to easily switch between Integration, Unit test, and Coverage projects.

• HTML Reports — All reports are generated in HTML by default, with user control over report layout and coloring. Customers who are interested in browsing through the release notes, or down-loading the production release should contact their sales person, please email to sales@vectorcast.com

All existing customers with a current maintenance contract will be sent keys for version 4.0 automatically.

About Vector Software

Vector Software, Inc. is a leading independent provider of automated test tools for software developers. Established in 1989 as a consulting and service organization, Vector's product focus is to empower software professionals to deliver the highest quality software in the least amount of time. Vector's "VectorCAST" line of products, reduce the burden placed on individual developers by automating and standardizing application component level testing. This innovative technology developed by Vector represents the "next generation" of intelligent embedded software test tools. The tools support Ada83/95, C/C++ and Embedded C++ (EC++). Over 150 customers use Vector Software's products for embedded software testing worldwide. The market focus of Vector is on companies performing embedded systems development for aerospace, military, medical, telecom, and process control related projects.

Vector Software's Product Family

VectorCAST/Ada

VectorCAST/C++

VectorCAST/RSP

VectorCAST/Cover

MC/DC Coverage for DO-178B Level A certification

DO-178B Qualification Packages

Ada and GNU/Linux

Debian Policy for Ada

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sat, 18 Mar 2006 02:15:29
Subject: Debian Policy for Ada, Second Edition
Newsgroups: comp.lang.ada

After more than a year, I have just uploaded the second edition of the Debian Policy for Ada. It is now available on Ada-France's web site as HTML, plain text, and PDF. In addition, the Debian .deb package contains an info version.

As I explained during the FOSDEM, the Policy itself has not changed. Basically, it consist of two points: (1) all Ada packages must use the same compiler from the gnat package, and (2) all libraries must follow the GNU Ada Environment Specification for the filesystem structure, and provide GNAT project files.

However, I have reworded most of the document for clarity, and updated a lot of time-dependent information. There are also details on the planned transition to GCC 4.1 in Etch in an appendix.

http://www.ada-france.org/debian/

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

And for your /etc/apt/sources.list, if you use Debian:

deb http://www.ada-france.org/debian/ ada main

deb-src deb http://www.ada-france.org/debian/ ada main

As always, I welcome comments and suggestions.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 28 Apr 2006 03:57:54
Subject: New home for the Debian Ada Policy
Newsgroups: comp.lang.ada

Some of you may have noticed the sad demise of the Ada-France web site; the server never recovered from a hard disk crash. That server was the place where I published the Debian Policy for Ada, and where I placed my Debian packages for my sponsors to pick up and upload to the official Debian archive.

Stéphane Richard has been kind enough to provide a Giga of disk space on his web site so I could publish these assets again. The new URL for the Debian Policy for Ada is thus:

http://www.adaworld.com/debian/debian-ada-policy.html

It is also possible to just browse the file hierarchy under

http://www.adaworld.com/debian

There you will find the Policy in plain text, info, PDF and the original Texinfo source formats, as well as a Debian repository with my latest and greatest packages. I periodically delete packages from that archive as they are uploaded into the official Debian archive.

Impatient Debian users can also add the following lines to their /etc/apt/sources.list:

deb http://www.adaworld.com/debian ada main

deb-src http://www.adaworld.com/debian ada main

The repository also supports pinning if you add the following to /etc/apt/preferences:

Package: *

Pin: release a=ada

Pin-Priority: 600

(adjust the Pin-Priority to taste). Then you can use "apt-get install" and "apt-get source" to obtain my packages before everyone else :) Otherwise, you can just wait a few more days and get the packages from Debian as usual.

Many thanks to Stéphane Richard for this web space.

I would encourage anyone interested to mirror this space so that the Debian Ada Policy is always available somewhere on the web.

[See also same topic in AUJ 25-3 (Sep 2004), p.126. — su]

Ada in Next Debian Release

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 7 Apr 2006 03:50:18
Subject: Ada in Debian: gnat-4.1 is now in testing
Newsgroups: comp.lang.ada

The package gnat-4.1 (=4.1.0-1) has been in unstable for 5 days and, since it didn't have any release-critical bugs, reached Debian Etch/testing today. It contains one preliminary patch by yours truly, but nothing yet related to Ada. It is now easier for Debian Etch users to experiment with gnat-4.1; just do:

$ apt-get install gnat-4.1

(beware: it conflicts with gnat, which is still the officially supported Ada compiler).

The GCC maintainer for Debian was waiting for GCC 4.1 to be in Etch before accepting my Ada-related patches into it. I already have a few patches ready, namely:

- support symbolic tracebacks

- look for project files in /usr/share/ada/adainclude by default

- link the GNAT tools dynamically against libgnat

- README.gnat

And I'm working on:

- build libgnatvsn and link the GNAT tools dynamically against it

The next step will be:

- build libgnatprj and link the GNAT tools dynamically against it.

I'll send these patches to debian-gcc at lists .debian dot org this weekend.

These patches will most probably appear in 4.1.0-2 in the next few days. If you're interested, you can look at the Debian build scripts and patches for GCC here:

http://svn.debian.org/wsvn/gcccvs/branches/sid/gcc-4.1/debian

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 03 May 2006 00:57:56
Subject: Ada in Debian: gcc-4.1 4.1.0-2 has reached unstable
Newsgroups: comp.lang.ada

The planned transition to GCC 4.1, outlined in the Debian Policy for Ada[1], is making progress. Today saw the upload of gcc-4.1 4.1.0-2, which includes my first batch of patches ported from gnat 3.15p. See the changelog[2] for a summary of these changes. With this upload, I have now ported all the changes I made in gnat to the newer gnat-4.1.

Of course, there are problems, since this is the "unstable" distribution. In particular, 4.1.0-2 failed to build from source[3] on the AMD64 and SPARC autobuilders[4] due to autoconf (which we in Ada-land all know and love), and might fail on other architectures too. I think the FTBFS issue will be fixed in the next few days. On the good side, the packages are already available as prebuilt binaries for i386, powerpc, and hppa. Yes, if you have one of 'em HP9000 boxes running Debian, you can now do Ada. Ada on a Superdome, anyone?

[1] http://www.adaworld.com/debian/debian-ada-policy.html

[2] http://packages.qa.debian.org/g/gcc-4.1/news/20060502T162915Z.html

[3] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=365780

[4] http://buildd.debian.org/build.php?pkg=gcc-4.1

Bug #365780 is release-critical because of its severity, "serious". Therefore, gcc-4.1 4.1.0-2 will not migrate to Etch/testing; perhaps 4.1.0-3 will, if it fixes that bug (and that's very likely). If you find other bugs, release-critical or not, please report them.

The binary packages of particular interest to us on c.l.a are:

gnat-4.1

libgnat-4.1

libgnatvsn-dev

libgnatvsn4.1

libgnatprj-dev

libgnatprj4.1

Looking forward

My next steps will be to port the other packages, starting with asis and glade, to gnat-4.1. Then I will package the latest versions of GtkAda and GPS, and the remaining packages after that will be piece of cake. When all that porting is done, I will then perform the official big-bang transition by uploading a new version of the package gcc-defaults so that the default Ada compiler becomes gnat-4.1. I anticipate that it will take another couple of months to complete the transition, and that we'll be well in time for the release of Etch in December this year.

Biarch is another major area that needs work. Several people have expressed interest on this newsgroup (hello Adrian). Here is a summary of the situation as I understand it; please correct me if I got anything wrong:

a) not even AdaCore support biarch

b) there is currently no platform that offers a biarch GNAT

c) this is uncharted territory

d) I don't (yet) have a 64-bit laptop to work on

e) I have no interest, other than intellectual, in biarch

As a consequence, if nobody helps me, then Debian will not provide biarch for Ada. Nor will any other operating system, for that matter, per b) above. I am more than willing to coordinate and provide advice, but I am not prepared to do the actual work. I think it is safer for Debian users if I concentrate on good uniarch packages first, and then look at biarch, if I have spare time, in the experimental distribution (i.e. it would not be in Etch). If, OTOH, several interested people band together and do a coordinated effort to bring biarch to life, then we can have it in Etch. Ideally, we'd need at least one person for i386/AMD64 and one person for PowerPC/PowerPC64.

For uniarch, perspectives are more rosy. I know that gnat-4.1 can be made to work on AMD64 and PowerPC64 with a little love and care. I don't know about other platforms, but since we now have hppa, anything is possible :)

If you would like to help, please visit http://lists.debian.org and send mail to debian-gcc@. Patches are greatly appreciated.

Oh, one last thing: after the transition, I'm planning to drop libcharles0 since gnat-4.1 provides Ada.Containers. If anyone wants to keep libcharles0 in Debian for compatibility reasons, please adopt the package.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 7 Apr 2006 05:02:56
Subject: Re: Ada in Debian: gnat-4.1 is now in testing
Newsgroups: comp.lang.ada

Someone asked me privately about 64-bit support in GtkAda, and I thought I'd clarify the situation once more here, as this information may be of general interest.

The transition to gnat-4.1 is in progress but still in early stages. Like I said, my first patches are not even public yet. I will try to provide 64-bit binary packages only after I'm done with the compiler, because a 64-bit compiler is a prerequisite for any 64-bit binary packages (d'oh). I'm not sure yet how much work is needed for good biarch support in gnat, but basically I want to make sure that both -m32 and -m64 work, and that libgnat has both 32-bit and 64-bit versions. Since I don't have 64-bit hardware, I cannot do this by myself, so I will rely on input from people who are sufficiently interested in biarch to invest some of their time into it. Or, maybe someone could sponsor me and buy me a 64-bit laptop? My workhorse is starting to show its age (it's an IBM ThinkPad T22 from 2001 with an Intel Pentium III @900 MHz — a bit slow to build GCC).

[See also "Multi-arch" in this issue —su]

Multi-arch

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 17 May 2006 02:22:57
Subject: Multi-arch
Newsgroups: comp.lang.ada

I am following up on the questions raised here about running both 32-bit and 64-bit binary code on the same machine. This is called biarch. Currently, the GCC-4.1 source package in Debian supports biarch for a few languages, but this support is maintenance-intensive and partial. Most notably, Ada does not currently support biarch (and neither does AdaCore). Furthermore, administration of a biarch system is more complex than that of a single-arch system, as not all binary packages support biarch, and some support packages are required, e.g. ia32-libs.

Canonical, the company behind Ubuntu (a derivative of Debian), has written two very interesting papers about multi-arch, including an introduction, problems, and a proposed solution. Multi-arch is a generalisation of biarch, and allows mixing packages for several architectures that are compatible with a processor. For example, an Athlon or Opteron system can run binaries for all of AMD64, i386, i486, i586 and i686. The proposed design would solve the current problems with biarch, at the expense of rewriting dpkg almost from scratch.

http://multiarch.alioth.debian.org/

[See also "Ada in Next Debian Release" in this issue —su]

From: Björn Persson <rombo.bjorn.persson@sverige.nu>
Date: Thu, 18 May 2006 21:24:44
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

I understand how multi-arch is useful if you want to use proprietary binaries that are available only for certain architectures. Otherwise I don't see a reason to do it. I'd think free code would just be compiled for the "native" architecture. Is multi-arch useful in an altogether free system in some way that eludes me?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Newsgroups: comp.lang.ada
Subject: Re: Multi-arch
Date: 19 May 2006 00:23:41

Yes.

Today, for example, OpenOffice works only on 32-bit architectures; if you want to run it on an AMD64 machine, you need biarch or multiarch. There are probably other cases where compiling for one of the architectures in a multi-arch system is problematic.

Also, 32-bit binaries use less memory than 64-bit binaries do. It makes sense to run 32-bit binaries on 64-bit machines, if those binaries do not benefit from the larger address space or 64-bit instructions.

Another benefit of multiarch is that you can e.g. compile i386 binaries on an AMD64 machine and run them on i386 machines.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sat, 20 May 2006 13:18:46
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

> That sounds like a defect in OpenOffice, so in that case multi-arch support functions as a workaround for a buggy program.

I think it's a bit more complex than that; it may well be a defect in OpenOffice, but also in the compiler or in any of the numerous libraries that OpenOffice uses. But I agree, in this case multi-arch is a workaround; this does not make multi-arch a bad idea, though.

> Ah. Yes, that's a good reason if the difference in memory usage is significant. (A factor two perhaps?)

A factor two for pointers and integers, yes, but not for Strings or other data structures. If you have many pointers, the increase in memory usage is quite significant.

> And then the compiler would also be an i386 program I presume. What are the benefits of that over an AMD64 to i386 cross compiler?

On the AMD64 machine, you need either an AMD64-to-i386 cross-compiler (gcc -m32 does just that), or a native i386 compiler. Either will do, but then you also need the i386 libraries to link against. A proper multi-arch design allows you to have these libraries alongside the AMD64 libraries, in a clean way as opposed to local hacks.

From: Dr. Adrian Wrigley <amtw@linuxchip.demon.co.uk.uk.uk>
Date: Sun, 21 May 2006 16:03:04
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

I suggest that people who run a mixture of machines may want to share binaries across the network, but build them on their newest machines.

People also want to avoid recompiling working code with another architecture, even if they have source code. You can't be sure it'll work exactly right without testing, which may be expensive. I found a couple of latent errors when building for 64-bit. One was calling a C varargs function incorrectly from Ada. Another was caused by undocumented members of a C struct overwriting the Ada stack. Changing architectures is a risk.

If you have written some code in assembly language, this will constrain the architecture until alternative code is available.

Finally, some people store records directly in data files for various reasons. Changing architecture would need recreation of those files.

It only takes one old architecture library to be a show-stopper, whether that is a third-party or handwritten codec, a buggy library or whatever.

multi-arch eases the transition to a new architecture.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Wed, 17 May 2006 11:45:19
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

IIRC, HP has always played with mounts per architecture. They also have an interesting file system standard. So I trust there is good reason to rewrite dpkg, given that only uni-arch is probably reasonably simple?

From: gshapovalov@gmail.com
Date: 17 May 2006 13:31:49
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

Yes, it can be done and is very well worth it. On Gentoo it is called multilib and is supported distribution-wide, not only for x86 and AMD64, but PowerPC and SPARC have theirs multilibs too IIRC (I

think both of them, but may be only SPARC actually). GCC is supported, as well as Ada and most of the libs and apps (to the point where you can have wine running 32-bit Windows code on otherwise 64bit-clean system — kernel and userspace).

This is regulated primarily via multilib and toolchain eclasses, which is basically the way to contain common controlling code. I recently put Ada on the same rails, so that now, on a multilib system, (that is for users who selected the multilib profile) two sets of rts libs are generated and compiler can generate 64 bit or 32 bit code. Although, as there were no requests so far, I haven't yet automatized switching between the multilib sub-profiles, only the usual niceties — like having FSF's 3.4.x, 4.1.x based and AdaCore's GNAT GPL compilers installed side-by-side and activated as necessary...

You can see the code here:

http://www.gentoo.org/cgi-bin/viewcvs.cgi/eclass/

You would be looking for the toolchain.eclass, multilib.eclass and possibly some other eclasses. And here:

http://www.gentoo.org/cgi-bin/viewcvs.cgi/dev-lang/gnat-gcc/

http://www.gentoo.org/cgi-bin/viewcvs.cgi/dev-lang/gnat-gpl/

The ebuilds and eclasses are mostly just a bash code, so should be familiar, aside from a few vars that have special meaning. Some Ada-specific implementation details and discussion can be found in this bug:

https://bugs.gentoo.org/show_bug.cgi?id=111340

and you can catch me on irc,freenode.net in channels #gentoo and #gentoo-dev if you would like to discuss this further (my nick there is georges, email of course works too: george at gentoo.org), although I suppose the way it will have to be done on Debian would be quite different...

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 18 May 2006 04:39:58
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

Interesting. I've looked at multilib.eclass and gnat-gcc-4.1.0.ebuild. I have no detailed knowledge of the Gentoo portage system, so I can only guess, but it seems to me that the current infrastructure means that each source package builds several binary packages, one for each arch, on the machine where you build. So, for example, if you have an AMD64 machine, you'd get two binaries, i386 and AMD64. Furthermore, the i386 package would install libraries in /usr/lib32, whereas the same package built on a i386 machine would install libraries in /usr/lib, so the two packages would be slightly different, and incompatible, even though built for the same architecture. Debian is currently in a similar situation, except that it has not deployed this scheme distribution-wide but only in a few important packages, binutils and gcc being the most prominent ones.

In the proposal, a source package would only produce one binary package for the machine doing the build; thus your AMD64 box would only produce the AMD64 binary. Then, if you want to install the i386 binary as well, you'd take the package built by the i386 autobuilder, modify it on the fly (this is one of the proposed changes to dpkg), and install it alongside your AMD64 package.

Of course, this scheme only makes sense in the context of a binary distribution like Debian, but there are several benefits:

- it reduces the workload of the autobuilders

- it reduces the size of the binary distribution, and load on the mirrors

- it simplifies system administration

- it simplifies the job of package maintainers and reduces the opportunities for bugs.

From what I understand, Gentoo people might not be very interested in these benefits, because:

- Gentoo has no autobuilders, as each user recompiles the world on their machine

- Gentoo has no binary distribution apart from the minimal bootstrapping system

- Gentoo users seem to like system administration :)

- Gentoo package maintainers seem to like difficult problems :)

The proposal also hints at the LSB. I think it would be necessary to standardise the library paths across all distros. The current /usr/lib, /usr/lib32, and /usr/lib64 directories are not general enough. Consider that some HP processors can run i386, AMD64, IA64, HPPA *and* HPPA64 binaries on the same machine :) And what about Cell processors and other future asymmetric multiprocessors? What about binaries intended to run on GPUs or other coprocessors?

From: gshapovalov@gmail.com
Date: 18 May 2006 09:34:34
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

This is actually quite similar to what is happening in Gentoo, as it does not make sense to make two versions of *every* package. Only the principal libs (parts of glibc and GCC RTS) plus compatibility libs (of these actually only the ones that are dependencies of requested packages) are produced "by default". Then user is free to mix and match in a usual fashion, although most people just stick with defaults of course :).

Although many people claim that their system administration efforts were reduced after switching to Gentoo :). Well, as usual, this is an issue of how you think and what tools you like I guess..

Definitely, [a LSB effort] will have to be done. Unfortunately I do not see it happening *just yet* — having seen how much it takes to organize anything on a large scale :). However in 2-3 years and when we can persuade LSB people that Linux/BSD/FOSS is not limited to Red Hat... (admittedly they are getting better at that lately). But we can start by having a discussion among Gentoo and Debian toolchain people.

From: Dr. Adrian Wrigley <amtw@linuxchip.demon.co.uk.uk.uk>
Date: Sun, 21 May 2006 13:05:25
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

Since I have been raising this as an issue in the past few months, I thought I'd give an update on my progress.

I am now running a prototype stock trading system on an AMD64 system running Fedora Core 5 (x86_64).

The problem I had been trying to solve was accessing very large memory-mapped market data arrays (several GB). I had run out of addressing range on IA32 systems.

Moving to AMD64, however, I found I couldn't link in commercial 32-bit library code, available in binary form only.

The system is based around Annex E distributed computing, with client partitions accessing server code using the 32-bit library via a Remote_Call_Interface partition.

I now build the system using AMD64 architecture for all the partitions except this Remote_Call_Interface partition, which is built for i386. There is no problem mixing architectures in a program as long as each partition only links in code from a permitted architecture. (Note that it will build invalid binaries with mixed architecture!)

The build process relies on a full i386 install of FC5 in a separate set of partitions from the x86_64. I installed a complete suite of i386 GNAT tools while running in 32-bit mode. The machine is dual-bootable, for convenience, but the build scripts use linux32 and chroot to compile 32-bit when booted as x86_64.

I have the same source code checked out from CVS in two separate directories. The build script checks the architecture, and if it is x86_64, it builds all the partitions except those which need the 32-bit library. Otherwise, it builds all the partitions.

I build one source tree in a 64 bit environment, the other in a 32-bit chrooted environment. I copy the 32-bit-only partition once built into the 64-bit build directory. This way, I get two complete working builds.

The GtkAda component 'gate' does not seem to work on x86_64. This is invoked by the build script in a 32-bit chroot.

I had experimented with -m32 in GNAT, and this produced working 32-bit .o files, but needed messing with at the linking stage, since it seemed to be looking in the wrong directories for the libraries. This might be made to work fairly easily, but I cannot get the -m32 switch passed from gnatdist as a compiler argument. This seems to be a simple bug(?).

I think having two separate build directories is necessary for building a mixture of architectures, even if each partition is needed once, since some .o files are needed in both architectures (for pure partitions, for example). I haven't yet arranged it to build the binaries in side directories from the same source files. This would help ensure a consistent code base.

I would like to thank Ludovic Brenta and Martin Krischik (and others) for working on these issues, providing helpful posts here at c.l.a, and for providing the necessary packages for running GNAT on these systems.

The 64-bit system is still at a development stage on a test machine. I may try Debian for the production system, but I think the same build process will be needed until the issues mention above are fixed.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sun, 21 May 2006 15:41:49
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

I guess you could have just one checkout, but two different object and executable directories. This is quite easy to set up in GNAT project files or, failing that, Makefiles.

I don't know if the proposal from Canonical will be accepted or implemented, much less in what time frame. What do you think I should do in Debian? Continue supporting single-arch only (which still allows you to use a chrooted system for the other arch, like you are doing now) and wait for dpkg 2.0, or try provide multi-arch GNAT and libraries now?

From: Dr. Adrian Wrigley <amtw@linuxchip.demon.co.uk.uk.uk>
Date: Sun, 21 May 2006 16:07:01
Subject: Re: Multi-arch
Newsgroups: comp.lang.ada

I found that I could build code with -m32 in FC5 with Martin's rpms, provided I messed around with linker commands and copied the 32-bit libraries from elsewhere. This would be OK, provided there was a "how to" document somewhere explaining it. It's what you call "local hacks" in another post.

But the lack of a functioning gnatdist -m32 option, and lack of 'gate' meant I had to use the chroot method. Using chroot took a lot of time to set up, with a whole new install and 'bind' for directories in fstab, and implementing a dchroot script, needing sudo etc. If gate and gnatdist were fixed, and the -m32 linking were documented, it wouldn't be worth the effort providing multi-arch GNAT libraries at this stage. It's only people with special requirements like mine who would really benefit.

Perhaps a year from now, once there are more AMD64 users, and once higher priorities are addressed, looking at the problem again would be a good idea.

I don't know enough about the amount and value of work needed on multi-arch GNAT to be able to be sure whether it makes sense. But I'm sure people value a streamlined robust, current and complete set of packages for each separate architecture.

Ada and Microsoft

AdaGIDE — Ada GUI IDE for Windows

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Mon, 20 Mar 2006 22:21:56
Subject: Ann: AdaGIDE 7.41 release
Newsgroups: comp.lang.ada

AdaGIDE (the Ada GUI Integrated Development Environment) is a lightweight but powerful interface to the GNAT compiler featuring a color context-sensitive editor and a code reformatter. It runs on Windows NT,2K,XP as well as the Windows 9x,ME series.

URL: http://adagide.martincarlisle.com

* Main improvements in AdaGIDE 7.41 compared to version 7.30

- Function to search through all open documents

- Possibility of running user-defined external tools on demand or automatically (check-in/out for source version control, call a code analyzer like AdaControl, run gnatelim, etc.)

Ada 2005 in Visual Studio 2005

From: Martin Carlisle <carlislem@acm.org>
Date: 3 May 2006 07:04:23
Subject: Ada 2005 in Visual Studio 2005
Newsgroups: comp.lang.ada

The A# compiler (Ada for .NET), a free GPL software product, has now been integrated into Visual Studio 2005. For more details, see http://asharp.martincarlisle.com

From: Martin Carlisle <carlislem@acm.org>
Date: 5 May 2006 07:43:48
Subject: Re: Ada 2005 in Visual Studio 2005
Newsgroups: comp.lang.ada

> Can we use the .NET environment coupled with A# for coding Ada 95 applications.

A# and Ada 2005 have the same syntax. You can write Ada 2005 (Ada 95) applications using A# and .NET.

From: Srini <RSVasan1007@gmail.com>
Date: 5 May 2006 04:51:13
Subject: Re: Ada 2005 in Visual Studio 2005
Newsgroups: comp.lang.ada

That is wonderful.

Will the "free" Visual Studio C# download available from Microsoft work? I am not sure what the version number of that is.

From: Martin Carlisle <carlislem@acm.org>
Date: 5 May 2006 07:45:05
Subject: Re: Ada 2005 in Visual Studio 2005
Newsgroups: comp.lang.ada

I do not believe the Express (free) version of C# will work, but I haven't tested this. The documentation from MS says you have to at least have the "Standard" version.

References to Publications

AdaCore Technical Papers

Author: Jamie Ayre
Date: Wednesday March 8, 2006
Title: Certification & Object Orientation: The New Ada Answer
URL: http://www.adacore.com/2006/03/08/certification-object-orientation-the-new-ada-answer/

The object model of Ada 2005 is well-suited for applications that have to meet certification at various levels. We review the use of Ada in the context of certification, and show that the object-oriented facilities of the current language standard, properly restricted to avoid dynamic dispatching, can already be used without problems under current DO-178B guidelines. We then examine the complications to certification that are presented by dynamic dispatching in a single inheritance model, and show implementation-specific ways of addressing these complications. Finally, we discuss the problems introduced by the use of multiple inheritance. We conclude by showing how, regardless of the extent to which object-oriented idioms are used, Ada provides a safe and efficient vehicle to create certifiable systems.

Author: Jamie Ayre
Date: Thursday March 30, 2006
Title: Safety, Security, and Object-Oriented Programming
URL: http://www.adacore.com/2006/03/30/safety-security-and-object-oriented-programming/

When safety-critical software malfunctions people lives are in danger. When security-critical software is cracked national security or economic activity may be at risk. As more and more software embraces object-oriented programming (OOP) safety-critical and security-critical projects feel compelled to use object-orientation. But what are the guarantees of OOP in terms of safety and security? Are the design goals of OOP aligned with those of safe and secure software (S3) systems? In the following sections we look at key OOP aspects and analyze some of the hazards they introduce with respect to S3 and outline a possible way of addressing these vulnerabilities. Specifically, after a quick overview of OOP in section 2, section 3 deals with inheritance and shows some of its hazards in terms of S3 along with possible remedies. Section 4 focuses on dynamic binding and suggests a safer and more secure implementation than what is conventionally done. Finally, section 5 looks at testing programs with dynamic binding.

[See also "Object-Oriented Programming and Safety" in this issue — su]

Ada Inside

Boeing 787 Air Conditioning Control Unit

URL: http://www.adacore.com/2006/05/01/hamilton-sundstrand-selects-gnat-pro-for-boeing-787-air-conditioning-pack-control-unit/

Monday May 1, 2006

Hamilton Sundstrand Selects GNAT Pro For Boeing 787 Air Conditioning Control Unit

SALT LAKE CITY, USA — Today at the Systems & Software Technology Conference AdaCore announced that Hamilton Sundstrand has chosen AdaCore's GNAT Pro as the Ada development environment for the software running in their Air Conditioning Pack airborne software configuration, which regulates cabin air temperature on the Boeing 787 aircraft. As part of the contract, AdaCore will adapt its flagship GNAT Pro Ada development environment to generate code for the 787 Pack Control Unit's designated MPC5554 microcontroller, and provide support for this new, specialized configuration.

The Boeing 787's Pack Control Unit is the first cabin air temperature control system to utilize Freescale's™ MPC5554 microcontroller (MCU). Hamilton Sundstrand chose the MPC5554 MCU because it offers a BookE compliant PowerPC™ core, a high level of integration, high system performance, reliability, real-time control and the ability to reuse legacy software architecture, reducing development cycle time.

"Implementing GNAT Pro for the PowerPC BookE compliant e200z6 core of the 5554 is very exciting for AdaCore," said Robert Dewar, President of AdaCore. "The MPC5554 is loaded with control functionality and is clearly attractive to designers of critical embedded systems for which Ada is the preferred programming language."

The processor is a variation on the standard AIM PowerPC architecture that AdaCore has been supporting for many years. The differences derive from the processor's incorporation into highly integrated embedded microcontrollers, and the port of the GNAT Pro toolset utilizes all of the work done for the PowerPC while taking advantage of the GCC compiler technology's flexibility to adapt to the different selection of instructions.

"This is the first GNAT Pro port targeting an MPC5554 processor, and we are looking forward to this opportunity," added Dewar. "Our advanced Ada environment will help Hamilton Sundstrand manage their development and exploit the advantages of Ada for mission-critical systems."

"We knew that finding an Ada compiler vendor that could design, deliver and support our designated hardware platform would be critical to the success of this project," said Andrew Wayner, Senior Software Designer at Hamilton Sundstrand. "We selected AdaCore based on previous technical expertise with both Ada compilation systems and with avionics application development environments, specifically on the A380 cabin air conditioning system project. The company also demonstrated a serious commitment to port GNAT Pro to the MPC5554 and see the project through to fruition."

About Boeing 787 Air Conditioning PCU

Two identical, dual channel, fully digital Pack Control Units (PCU) provide most of the control and monitoring of the Cabin Air Conditioning and Temperature Control System (CACTCS) on the Boeing 787. The CACTCS provides cabin heating and cooling for passenger, crew and cargo zones utilizing two air cycle packs, each controlled by a PCU. Each channel of a PCU contains a Freescale MPC5554, which provides a plethora of features such as CAN communications, Queued Analog-to-Digital Converters and an Enhanced Modular Input/Output Subsystem. The PCU takes advantage of these features to acquire data from a multitude of sensor types, including pressure and temperature as well as providing accurate and real-time control of its motor control outputs.

About Hamilton Sundstrand

Hamilton Sundstrand, a United Technologies Company, is headquartered in Windsor Locks, Connecticut, and manufactures and services advanced technology aerospace and industrial systems. It employs approximately 16,000 people worldwide. United Technologies, based in Hartford, Conn., is a diversified company that provides high-technology products and services to the aerospace and commercial building industries.

Indirect Information on Ada Usage

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

(...) Work in high-integrity programming language design and static verification. You will form part of this team, working in all areas of the product lifecycle including R&D, customer support, sales, delivery of training, and marketing.

(...) You will require a robust set of technical skills and have experience in one or more of the following: programming language design, static analysis tools or compilers, theorem proving, software model checking, SAT solving or grid-based computing. You will probably have at least a 2:1 degree in a related subject. You'll be a strong team member, have the ability to think analytically and have clear customer focus. Applications from recent graduates or those with higher degrees are welcome.

(...) For our aeronautics skills centre, we are currently looking to recruit Ada 83 or 95 SOFTWARE ENGINEERS specialised in ADA DESIGN and DEVELOPMENT (m/f). Experienced in ADA DESIGN, we propose you to join our team working on most challenging projects, in the highest speed area (space and avionics) and the railway sector. Knowledge in aeronautic or military standards or railway standards is a plus.

Industrial Engineer (Ing) or Civil Engineer (IR) with good knowledge & experience in Ada 83/95 (min 3 years). You have very good communication skills and English is mandatory as the development is done on a very international and multi-site basis, with frequent meetings and close interactions.

Ada in Context

Java-like Exception Contracts

From: Maciej Sobczak <maciej@msobczak.com>
Organization: CERN — European Laboratory for Particle Physics
Date: Wed, 22 Mar 2006 10:24:10
Subject: Re: Handling invalid objects
Newsgroups: comp.lang.ada

Dmitry A. Kazakov wrote:

> Still exception contracts would greatly improve safety of Ada as a language.

If by exception contracts you mean embedding the exception specification in the "signature" of the procedure/function, then it was already exercised by the Java community with rather disappointing effects.

From: Maciej Sobczak <maciej@msobczak.com>
Organization: CERN — European Laboratory for Particle Physics
Date: Wed, 22 Mar 2006 17:42:53
Subject: Re: Handling invalid objects
Newsgroups: comp.lang.ada

> If Java did it wrong, let's do it right in Ada.

Do what exactly? This is important question. The problem with exception specifications is that they are self-contradictory:

- We use exceptions when we want to *DECOUPLE* error reporting from error handling. We find it especially good in those situations, where error reporting site and error handling site are separated by more than one level of subroutine calls (otherwise returning error codes is good enough).

- We embed contract information in subroutine signatures to *COUPLE* the caller with the callee with respect to what they provide to each other and what they expect from each other.

Now, "coupling" and "decoupling" are hardly compatible. Let's see where it breaks in so-called practice.

First, there is a cascading effect when someone on one end of the chain adds a new exception type. Just let's say that the project evolved and for example a database got involved in something that was previously managed with the use of files. There is a new DBError exception, possibly having some db-specific information encoded (you know, exceptions are real objects in some languages :)). This error is not handled neither by the offending function nor anybody in the chain, but is supposed to be handled at some higher level. In this scheme, the poor programmer has to add the DBError type to the exception specification to *all* functions in the chain. And apart from being a maintenance horror, it might be just impossible because the functions on the road are already closed or just owned by someone else. The tempting "solution" is to shut up the exception to meet the specification which was already cast in stone. Just grep any bigger Java project for things like:

```
catch (Throwable e) {}
  // <- empty block here!
```

to see it at work.

Java guys can at least try to fight this problem with inheritance. The exception need not be exactly of the specified type, but might be something derived from what was specified. So, the other temptation is to specify the exception type that is rather general (higher in the inheritance hierarchy) to ease the accommodation of new exception types. But the more general is the specification, the less useful it is with regard to enforcing anything. In the extreme, it does not enforce anything at all.

(Note that Ada would not have this possibility, or it would need to allow for exception hierarchies.)

Second, the problem is that the error reporting and handling might be stated as a contract not between immediate caller and callee, but between some entities that operate across some other entity. A C++ example could be:

```
void myFun()
{
    vector<MyType> array;
    // ...
    sort(array.begin(), array.end(),
         myComparator());
    // ...
}
```

Above, the call chain is myFun->sort->myComparator, but the error handling and reporting is the business which is agreed between myFun and myComparator only. The sort algorithm was written long before myFun and myComparator and it (sort) does not care what it sorts and for whom. This means that sort should be completely transparent to the agreement that myFun and myComparator might have with regard to error reporting and handling.

The solution might be to allow the compiler to synthesise the exception specs for sort automatically. But then, the specs would be just useless, because it would not enforce anything.

The real problem is that languages based on simple subroutine calls are not appropriate for expressing these kind of relationships. Something fundamentally different would be needed to ensure that myFun handles exceptions from myComparator without involving sort in this process, but I don't see what that thing would look like.

Anyway. The whole purpose of exceptions is to provide a kind of "out-of-band" channel which is *decoupled* from the main chain of subroutine calls. This "out-of-band" property is something that you either like (and/or accept) or not in the given project. Messing around with things like exception specifications is just swimming upstream - you cannot provide coupling into something that was *intended* to be "out-of-band" in the first place.

Java guys failed with this exercise. C++ community dropped the idea altogether before failing (C++ never had compile-time enforcements of exceptions specs and today nobody's using them anyway). I don't see how Ada would do something like this without incurring effects described above or without fundamentally changing something in the way subroutines are used. But I'm looking forward to see your opinions on this (and maybe learn something about Ada culture? :)).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 23 Mar 2006 14:20:24
Subject: Re: Handling invalid objects
Newsgroups: comp.lang.ada

No, we are decoupling using contracts. Instead of presenting any concrete caller, we do a contract. The callee is coupled only to its contract. It does not to any caller, because it hopes that any caller will respect the contract.

This [cascading effect problem] is not specific to exceptions. It is "fragile class" design.

You cannot add a new exception type [better to say a class of], this breaks the contract. You have to stay within the class.

In Ada model, where exceptions are values, this means that the exception contracts should specify ranges of values [subtype] and a new exception [value] should be chosen from that range. It is doable.

When you have some procedure composed out of another procedure, in this example, you pass it as a parameter, you could say something like:

A raises this plus anything what B does.

Because B has a defined subroutine type, its contract is statically known.

One could also bind exceptions to types of primitive subprograms. For example:

```
type File is tagged ...;
subtype File_Error is
   File'Exception;
--    The range of exceptions
--+  bound to File
procedure Read (X : File)
   exception File_Error;
type DB is new File with ...;
DB_Error : File_Error :=  some sugar;
--   Declares a new exception
--+ in the range
procedure Read (X : DB);
-- This is allowed to raise DB_Error
```

Exceptions allow us to weaken preconditions. Without exceptions, a real-valued sqrt should specify x >= 0.0 as a precondition. This is unacceptable when x is statically unknown. Exceptions relax the precondition and bring things back to static. The price is that you leave the realm of real numbers. You have to this way or another. Either you make it complex-valued or you say that the result is "Real or Constraint_Error." Who will deal with this result is the question for another day. But when exceptions are not contracted, then the gain of static preconditions gets lost. So in my view, Java's is undoubtedly right here.

From: Stefan Lucks <lucks@th.informatik.uni-mannheim.de>
Organization: Th. Informatik, Univ. Mannheim, Germany
Date: Wed, 22 Mar 2006 19:06:57
Subject: Re: Handling invalid objects
Newsgroups: comp.lang.ada

One thing Ada could reasonably do is to *enable* subroutines to *promise* to raise no exceptions, or only certain exceptions. (And, of course, to enable the compiler to verify if this promise is kept. This would, be a little bit similar to SPARK, which can prove the exception-freedom of subroutines.)

Of course, the implication is that the subroutine itself may only use (or rather "with" ;-) subroutines which make a similar promise, or have to handle all exceptions ("others").

As an example for a notation, consider the following subroutines which any freedom to raise and propagate exceptions deliberately:

```
function "+"(A, B: T) return T;
procedure Get (Item: out T);
procedure Put (Item: T);
```

The remaining source code is not Ada. (Or perhaps it is Ada 2015? :-)

No Exceptions raised:

```
function "+"(A, B: T) return T
   raise null;
procedure Get (Item: out T)
   raise null;
procedure Put (Item: T)
   raise null;
```

Some Exceptions may be raised:

```
function "+"(A, B: T) return T
   raise Constraint_Error,
         Program_Error;
--    can raise or propagate
--+ Program_Error, but nothing else

procedure Get (Item: out T)
   raise Ada.Text_IO.End_Error,
         Ada.Text_IO.Data_Error,
         Ada.Text_IO.Mode_Error,
         Ada.Text_IO.Layout_Error;
--   can raise or propagate these
--+ four exception, none else

Line_Failed : exception;

procedure Put_Line (Item: T)
   raise Ada.Text_IO.End_Error,
         Ada.Text_IO.Data_Error,
         Line_Failed;
--   can raise or propagate these
--+ three exceptions none else

procedure Put (Item: T)
   raise Ada.Text_IO.End_Error,
         Ada.Text_IO.Data_Error,
         package;
--   can raise or propagate
--+ two exceptions from Ada.Text_IO
--+ and any exception defined in the
--+ current package
```

I could also imagine a package to specify which errors might be raised or propagated in any of its subroutines. This would simplify notation. Consider the following almost-complete example for a package specification

```
with Ada.Text_IO;
package Some_Library
   raise Ada.Text_IO.End_Error,
         Ada.Text_IO.Data_Error,
         Ada.Text_IO.Mode_Error,
         Ada.Text_IO.Layout_Error,
         Constraint_Error,
         Program_Error,
         package;
--   Any subroutine defined here
--+ may raise the four exceptions
--+ from
--+ Ada.Text_IO, the two exceptions
--+ Constraint_Error, Program_Error,
--+ from Standard, and the
--+ exception(s) defined in the
--+ package, namely
--+ Line_Failed.

   Line_Failed : exception;
   type T is private;
   function "+"(A, B: T) return T;
   procedure PutLine (Item: T);
   procedure Put (Item: T);

private
   type T is ...;
   -- which type T wold you like?

end Some_Library;
```

Further, when a subroutines X formal parameter is access-to-subroutine, then any exception raised by a subroutine given as an actual parameter need not be handled by X. This should be the caller's duty.

Uninitialized variables in Ada

From: Brian May <bam@snoopy.apana.org.au>
Date: Tue, 14 Mar 2006 18:44:09
Subject: Re: private types
Newsgroups: comp.lang.ada

(...) some languages will initialise all variables to dummy values — this means you can get predictable results in code that (wrongly) uses them before setting them to a value.

In this case it is possible to force initialisation (at least outside the package), as per another poster's suggestion, because it is a private type. Alternatively it is possible to turn it into a record type and provide a default value for the component, as per another post.

In other cases it isn't so easy, e.g. any non-private non-record type.

In fact, by default (at least last time I checked), GCC (or was that GNAT) doesn't check the validity of parameters to functions if the type matches, even though the type hasn't been initialised and may just happen to contain an illegal value.

I seem to recall Ada will initialise access types to null, and record components (if defaults given), but nothing else.

From: Justin Gombos <usenet.ada.jog@xoxy.net>
Date: Fri, 17 Mar 2006 04:33:15
Subject: Re: private types
Newsgroups: comp.lang.ada

Beyond access types, I would not consider that feature you're describing helpful. In fact, it's more of a disservice.

The first problem: initializing to zero, or some other "dummy" value of the compilers choice is likely to result in a valid value (sometimes), which only serves to /hide/ bugs in the cases where the object is used prior to a meaningful assignment.

Then problem with user forced initialization (which is what the OP is after): it could mask the cases where reassignment is inevitable. IOW, suppose you have subprograms like this:

```
function Exists return Boolean is
--    Later assignment to Found_It is
--+ evitable
  Found_It : Boolean := False;
begin
  if Some_Precondition then
    Found_It :=
      Some_Other_Condition;
  end if;
  return Found_It;
end Exists;
```

In the above case, an initial value may persist if some path is not executed. The maintainer can immediately expect this to be the case upon seeing the initialization (assuming the author was competent). In other cases, an initial value may get overwritten no matter what. In these cases it makes more sense not to initialize, because it clarifies to the maintainer what kind of logic to expect before even looking at the body of code.

It's always irritating to be reading someone else's code, and find that they've blanket initialized objects needlessly. It hides bugs, and also obscures the logic from the maintainer.

We don't know enough about the OPs case to know whether forced initialization is wise, but he should be cautioned not to take this approach arbitrarily, or on a regular basis. It really depends on the situation.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 17 Mar 2006 19:17:32
Subject: Re: private types
Newsgroups: comp.lang.ada

Brian May wrote:

> For testing the code, as Found_It is undefined in the second test, it is possible it might just fluke the tests you give it and pass everyone.
> The first code is predictable though, and as long as you give it the same inputs, it will always produce the same outputs, making it easier (IMHO) to test.

It's not just testing. Ada 95 is very clear that an Ada compiler cannot assume an object is in range unless it can prove it is initialized. Explicit initialization makes this proof trivial (and leaving it out may make it impossible to prove.) Thus, given

```
A : Positive := 10;
B : Positive;
```

the compiler can assume that A is in range, potentially being able to eliminate checks and speeding up the code. But it cannot assume that B is in range (unless it can prove that it is initialized further on).

So I recommend initializing everything (or assigning it immediately after the begin) that could be significant to performance.

From: Justin Gombos <usenet.ada.jog@xoxy.net>
Date: Sat, 18 Mar 2006 02:17:12
Subject: Re: private types
Newsgroups: comp.lang.ada

As a rule, I try to put readability ahead of optimizations. But if I did want to write optimum code, I'm not seeing your point here.

The runtime checks that might be placed on B need not affect code not handling B. Assuming an extreme case, suppose B is not assigned until 100 lines later (i.e. not immediately following the begin). There should be no runtime checks in those 100 lines between the 'begin' and the first assignment to B if B is not referenced (and if B is referenced prior to assignment, that's a problem that outweighs excessive checks anyway). The first occurrence of B is going to be an assignment to B, and it must have the same checks that A would have if A were being reassigned at this point. So I'm not seeing why more runtime checks would occur in the case of B.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 20 Mar 2006 18:08:25
Subject: Re: private types
Newsgroups: comp.lang.ada

Because, in general, you don't know whether B is initialized. And Ada 95 requires that invalid values be detected before they cause any damage (with some unfortunate exceptions). If B is used to index an array, for instance, it must be checked unless the compiler can prove that it is valid. But that is very hard in general, because of path issues:

```
  B : Positive;
begin
  if Bafflegab (10) then
    B := 10;
  end if;
  ... Str (B) ...  -- Must check for
                   --+ invalid values here.
end;
```

There is no way that the compiler can tell if B has been initialized or not. And Ada 95 does not allow *assuming* that it is initialized (which is essentially what your argument boils down to) — the compiler must presume the program is incorrect for this purpose unless it can prove that it is not.

But note Bob Duff's point that there are other ways to arrange code generators that might have different effects on checking. That's true in general, but in this case in particular, the compiler cannot remove the check for Str (B) no matter what the code generation scheme. If B had been initialized, it would have been able to in most schemes.

In any case, in most real code, it's hard to prove something is initialized unless it is done right at the top. Moreover, compilers vary in the amount of flow analysis that they do. So preinitialization is the way to go for maximum portability. (But I suggest this when you're going to initialize the value anyway, as opposed to initializing it just for this purpose.)

You're right about premature optimizations, of course.

From: Dirk Craeynest <dirk@apollo.cs.kuleuven.ac.be>
Organization: Ada-Belgium, c/o Dept. of Computer Science, K.U.Leuven
Date: 18 Mar 2006 09:39:56
Subject: Uninitialized variables (was: Re: private types)
Summary: If you use GNAT, use pragma Initialize_Scalars.
Newsgroups: comp.lang.ada

We did (and do) feel [initializing everything] is not a good approach, at least not when using GNAT or another compiler that supports something like the pragma Initialize_Scalars and enhanced validity checking.

For much more about uninitialized variables in Ada code, the following paper might be useful:

[1] "Exposing Uninitialized Variables: Strengthening and Extending Run-Time Checks in Ada", Robert Dewar, Olivier Hainque, Dirk Craeynest, and Philippe Waroquiers, In [2] "Proceedings of the 7th International Conference on Reliable Software Technologies — Ada-Europe 2002", Vienna, Austria, June 17–21, 2002, Johan Blieberger and Alfred Strohmeier (Eds.), volume 2361 of Lecture Notes in Computer Science, pages 193–204, Springer-Verlag, 2002.

The conclusion of that paper contains our recommendation:

---start-quote---

5.3 Impact of Usage of Initialize Scalars on How to Program

There is a trend in programming guidelines to "force" initializing everything at declaration resulting in code like:

```
B : Natural := 0;
if .... then
  B := 5;
else
  B := 8;
end if;
```

The difficulty with such an approach is that the initial value is meaningless. If this value is used accidentally, the results are potentially just as wrong as the use of an uninitialized value, and furthermore, the explicit initialization precludes the approach we have described in this paper, and thus may introduce bugs that are much harder to find and fix. The automatic initialization under control of

the compiler using Initialize Scalars is a far preferable approach.

We therefore recommend that when a scalar is declared, the programmer should avoid initializing it if the code is supposed to set the value on all paths. It is better to let Initialize Scalars + gnatVa detect the bug in the code logic rather than trying to deal with meaningless initial values. Even for safety-critical programs, we can first compile with Initialize Scalars + gnatVa + invalid values and then, if needed, field the code with Initialize Scalars + all zero values (if it is the case that zero values give the code a better chance of avoiding seriously improper behavior).

---end-quote---

The GNAT manuals provide more information on GNAT's pragma Initialize_Scalars [3] and on enhanced validity checking [4]. Reference [3] mentions:

---start-quote---

Note that pragma Initialize_Scalars is particularly useful in conjunction with the enhanced validity checking that is now provided in GNAT, which checks for invalid values under more conditions. Using this feature (see description of the -gnatV flag in the users guide) in conjunction with pragma Initialize_Scalars provides a powerful new tool to assist in the detection of problems caused by uninitialized variables.

---end-quote---

We can assure everyone that from a developers and testers point of view the combination of Initialize_Scalars and enhanced validity checking is indeed "particularly useful".

References:

[1] <http://www.cs.kuleuven.be/~dirk/papers/ae02cfmu-paper.pdf>

[2] <http://www.springer.de/cgi/svcat/search_book.pl?isbn=3-540-43784-3>

[3] <http://www.adacore.com/wp-content/files/auto_update/gnat-unw-docs/html/gnat_rm_2.html#SEC48>

[4] <http://www.adacore.com/wp-content/files/auto_update/gnat-unw-docs/html/gnat_ugn_4.html#SEC47>

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 20 Mar 2006 18:38:40
Subject: Re: Uninitialized variables (was: Re: private types)
Newsgroups: comp.lang.ada

I disagree in detail with your conclusions, but probably not in general.

1) Initialize_Scalars is an Annex H thing that is rarely available in Ada implementations. GNAT is the only one that I know of that has it. I don't think offering advice that most users can't follow is very helpful.

2) Initialized_Scalars does no good when you have full range types (which are very common in a compiler, for instance). In that case, it is equivalent to initializing to a random value, and worse, it gives a false sense of security.

3) "The initial value is meaningless". Here I agree and disagree with you. The agreement is that you shouldn't initialize to a meaningless value. The disagreement is that for most variables, there is an obvious initial value (like Null for access types) that is not meaningless. For instance, I have a lot of string processing code in the spam filter that have length variables. I usually initialize the length to zero (empty), because that *is* the initial state of the object. So, much of time there is a useful initialization.

It think it is better to *avoid* uninitialized variables than to argue about how to *handle* uninitialized values. The example you gave:

```
B : Natural := 0;
if ... then
   B := 5;
else
   B := 8;
end if;
```

is awful, I agree. But I'd probably write:

```
B : Natural := 8;
if ... then
   B := 5;
   -- else use the default values
end if;
```

instead, and the initial value is no longer meaningless. Similarly, I use a lot of blocks, and try to keep the declarations on variables to scopes where their initial values are known (or immediately initialized). Both of these are better than *any* technique to handle uninitialized variables.

4) As your note suggested, assuming that everything is tested is dangerous. It's necessary in the fielded system to protect against uninitialized variables causing weird results. I just prefer to do it from the beginning (by reducing them as much as possible). And I'd prefer to rely on compile-time warnings (which GNAT also does well, BTW) to get rid of them at the source.

5) Any extra cost from initializing objects to meaningful values early (and such cost is usually quite small) will quickly pay for itself. (I think that is in line with the conclusions of the paper, too).

Conclusion: don't write uninitialized variables in the first place; but use your head to eliminate them — junk initializations are no better than the uninitialized variables that they replace. Mindless following of coding standards always produces junky code.

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Sat, 18 Mar 2006 15:06:19
Subject: Re: Uninitialized variables (was: Re: private types)
Newsgroups: comp.lang.ada

Here (trying to to sum up), three problems I see with the tactic of initializing everything:

- useless initializations (i.e. dummy values rewritten later) take time and usually _hurt_ performance (think to number crunching with huge objects, or frequently used functions with local variables)

- useless initializations introduce meaningless code lines

- useless initializations prevent detecting bugs that can be detected without these initializations (they can be detected by combining the Initialize_Scalars pragma and the validity checks)

My rule is rather to initialize *only* variables you can give a meaningful values. Of course it depends on the context. Maybe there are situations where you can prefer a program giving wrong results than an unhandled exception popping at the wrong moment (although I find the method very shocking!)... In such cases the systematic initialization could be a way (not nice but pragmatic) to silently disable bug detection.

From: Jeffrey Creem <jeff@thecreems.com>
Date: Sat, 18 Mar 2006 09:36:24
Subject: Re: Uninitialized variables
Newsgroups: comp.lang.ada

This is also the approach I follow. It has the added benefit that some compilers can now give you warnings about reading from it before you assign to it and thus help you find the bug. If one does the typical initialize everything to 0 or 'First or something like that then you can expect no help from the compiler.

Now in reality, compilers vary in their ability to provide useful warnings in this area.

GNAT does a reasonably good job of balancing real warnings in this case against false warnings.

Another compiler I use takes a different approach where it seems to warn in a lot more cases and thus ends up with a lot more false positives. It is probably not that bad of an approach if you used this compiler from the beginning but with lots of legacy code the signal to noise

Consider the following toy code:

```
with Text_IO;
procedure Toy is
   I : Integer;
   I_Set : Boolean := False;
```

```
   Should_We_Set_I : Character;
   J : Integer;
begin
   Text_IO.Get(Should_We_Set_I);
   if Should_We_Set_I = 'y' then
      I_Set := True;
      I := 1;
   end if;
   if I_Set then
      Text_IO.
        Put_Line(Integer'Image(I));
      -- This is ok
      Text_IO.
        Put_Line(Integer'Image(J));
      -- This is bad
   end if;
   J := 1;
end Toy;
```

GNAT warns on the line that says "This is bad" but not on the line that says this is ok.

Another compiler I use warns on both (Not posting other compiler here only because I have not tested this exact code on it and am making this assertion based on similar real code).

Obviously, unless one uses someting like polyspace, a simple compiler can't be expected to detect all of these path flow type cases.

The important point here is that if one had a convention that all variables should be initialized, no compiler could tell you that you were doing something wrong on the "this is bad line".

From: Martin Dowie
<martin.dowie@btopenworld.com>
Date: Sat, 18 Mar 2006 12:06:37
Subject: Re: private types
Newsgroups: comp.lang.ada

Or use a tool like PolySpace, which is very good at spotting this sort of thing. (www.polyspace.com).

From: Robert A Duff
<bobduff@shell01.TheWorld.com>
Date: 18 Mar 2006 07:47:09
Subject: Re: private types
Newsgroups: comp.lang.ada

This is not quite true. What the compiler can prove depends on the compiler's code generation strategy. Example:

```
type Index is range 1..10;
type A is array(Index) of Character;
X: Index; -- not initialized here
procedure P(Y: Index) is
begin
   ...
end P;
... -- (*) might initialize X here
P(X);
```

Suppose the compiler cannot prove that the code marked "-- (*)" will initialize X. The compiler has a choice: It can do a range check at the call to P, and then assume inside the body of P that Y is in range (even though the value _might_ have come from an uninitialized variable). Or, the compiler can avoid the range check on the call to P, in which case it cannot assume that Y is in range.

[Whether initializing everything could be significant to performance] also depends on the compiler. Many compilers can prove that a variable is initialized here:

```
begin
   if ... then
     A := 3;
   else
     A := 4;
   end if;
   ... --  Here, we can presume A
     --+ is in range.
```

Adding "A := 0;" between "begin" and "if" would be overkill for such compilers.

From: Justin Gombos
<usenet.ada.jog@xoxy.net>
Date: Fri, 17 Mar 2006 23:44:08
Subject: Re: private types
Newsgroups: comp.lang.ada

Robert A Duff wrote:

> I'm not sure what the right answer is, but surely all the arguments for and against dummy values apply equally to access types.

I don't agree with that. Null is a standard abnormal object for access types in all languages, and can never be taken for something valid. Null pointers are quickly detected, and easily understood.

But with any other type, null (zero) is most likely a valid value. This is probably why the ARM states:

The implicit initial value for an access subtype is the null value of the access type.

But makes no such rule for other types.

From: Robert A Duff
<bobduff@shell01.TheWorld.com>
Date: 18 Mar 2006 07:56:45
Subject: Re: private types
Newsgroups: comp.lang.ada

Null is neither "abnormal" nor "invalid" in Ada. As for "all languages", some have a concept of "null" or "nil" or whatever that is the same in this regard. Some languages have no such concept.

In Ada, if an object of an access type has no explicit initial value, you can't easily tell whether that means "null is a meaningful value for this variable, and that's the default I want" versus "this variable will be initialized to a meaningful (non-null) value later".

This is exactly analogous to the case with integers — if they were default-initialized to zero, you can't easily tell whether zero is intended as a meaningful initial value, versus later initialization to a meaningful value.

From: Robert A Duff
<bobduff@shell01.TheWorld.com>
Date: 19 Mar 2006 13:15:03
Subject: Re: private types
Newsgroups: comp.lang.ada

> > Zero has a universal meaning with access types, but it could be in range or out of range for any other type. The ARM selects access types specifically to get a default initialization of zero for this reason.

> Who says that Null := 16#0#? I could image a Hardware/CPU/OS where it would be better to define Null := 16#FFFF_FFFF_FFFF_FFFF#.

The convention on TOPS-20 for null (in all the various languages that have it) is to use some address other than zero. I don't remember which address. The OS protects that page so it traps (just as most modern operating systems do for page zero).

There was even an Ada compiler for TOPS-20, and of course it obeyed that convention.

Using all-zero-bits for null has some minor efficiency advantages.

> For example an OS who's Virtual Memory Management System assign address 16#0# to be a valid address and to hold some important process data to which the process needs access.
> Of course programming C or C++ on such an OS could be quite challenging (Write to (void*)0 and you mess up your Process Information Descriptor).

I believe the C++ rule is that 0 (written in your program) is the same thing as NULL — but it need not be represented internally by all-zero-bits. Casting the integer zero to a pointer, however, does not necessarily result in NULL. That's sort of confusing, but if you understand the rules, the "quite challenging" comment above does not hold.

From: Dr. Adrian Wrigley
<amtw@linuxchip.demon.co.uk.uk.uk>
Date: Sun, 19 Mar 2006 20:43:42
Subject: Re: private types
Newsgroups: comp.lang.ada

> Super! And how many (in %) of C++ programmer actually know that. By guess is 0.1%. And indeed I is the main problem: Only a very few C/C++ programmers actually master the language.

It is (or was) quite a common interview question, to see if C (and C++) programmers know their subject properly. As you say, Robert, most don't.

A related topic is the issue of pointer representation, which (IIRC) says that pointers to char (any kind) and void have to be the same. Pointers to functions have

to be the same. But all other pointers can have their own representation. All pointers can be converted to and from pointers to void, without loss. And pointers can have different sizes and different patterns for the null pointer. I suspect a lot of code would fail if compilers wanted to exercise their full freedoms!

Ada and the Open Source Movement

From: Jeffrey Creem <jeff@thecreems.com>
Date: Sat, 06 May 2006 15:55:49
Subject: Re: Why C for the Open Source Movement?
Newsgroups: comp.lang.ada

> Can someone explain me the reasons why the main actors of the Open Source community didn't choose Ada instead of C to write the core elements of their systems (Linux, Hurd, FreeBSD, etc...) and most of the critical applications that come with it if the main goal was to provide a real alternative to the "proprietary clan". Would it have reduce significantly bug list for all development projects and assure much more reliable applications? Isn't the Unix tradition based on well crafted design mechanisms? Does its traditional ways became so heavy to prevent adaptation and or renewal of its philosophy through time?

All of those projects started before there was a freely available Ada compiler.

In the case of someting like the GNU project itself, when Stallman started the GNU project, Ada 83 was not even really out yet and certainly not in a position where it was stable.

There is a lot of inertia to overcome once one starts an OS project. Switching the Linux kernel or Hurd or anything else of that scale to Ada just for the sake of the better language is probably not the best idea.

Ignoring all of the forking and grubling that it would create for a moment at the very least one would have to agree that it could be years before you got back to the same level of stability as the original kernel.

One could argue that it could be done piecewise but the problem with that is if you start small now you have to justify pulling in some small Ada runtime into the kernel just to support some particular module and again it does not seem to make sense.

There are lots of "silver bullet" languages out there that have cool features (e.g. Python) but still no one has tried to port the kernel to python.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 8 May 2006 02:05:26
Subject: Re: Why C for the Open Source Movement?
Newsgroups: comp.lang.ada

I think C was well suited as a bootstrap language, because C was designed to make the compiler writer's job easy. When starting the GNU project, RMS first wrote a C compiler, and then used that to write a Lisp system, which he used for Emacs. That was the right thing to do, IMHO.

Early on, Lisp was therefore in a much better position than Ada to become the language of choice for GNU. Even today, there is still quite a lot of Lisp in GNU/Linux systems; and not just in Emacs and XEmacs, as there are several free Common Lisp and Scheme systems available now. But even then, Lisp is still a minority language. So the OP's question remains valid, but in the context of the early days of the GNU project, I would ask "why C and not Lisp"? I think it may be because Lisp has or can be perceived to have too much overhead for writing libraries or a kernel, and so was summarily and incorrectly dismissed by application programmers. I would think the folks on comp.lang.lisp have already discussed this question many times.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 8 May 2006 02:13:22
Subject: Re: Why C for the Open Source Movement?
Newsgroups: comp.lang.ada

> Isn't the Unix tradition based on well crafted design mecanisms?

No, it's not. It started as a hack made by a few long-haired, rebel programmers in their spare time, bazaar-style. In contrast, MULTICS was intended to be a beautiful cathedral of software, and I think it was written in PL/I not C. MULTICS is still not complete, and I think it'll take as long to complete as it took actual cathedrals :)

But history showed that "worse is better", unfortunately for purists.

http://en.wikipedia.org/wiki/Worse_is_better

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 8 May 2006 15:12:11
Subject: Re: Why C for the Open Source Movement?
Newsgroups: comp.lang.ada

I don't think that it was simplicity. Unix never was simple, rather it was simply bad. (:-))

P.S. It is illogical to express badness in a positive way, like "simplicity", for example. If simplicity is the goal, then it is good, and thus bad cannot be simple! In fact, to be really bad (as Unix, or Windows is) is much harder than to be any good. So many things can be considered positive... It is very difficult to ensure that most combinations of them were indeed wrong... (:-))

From: thvv <thvv64@gmail.com>
Date: 6 Jun 2006 07:24:16
Subject: Re: Why C for the Open Source Movement?
Newsgroups: comp.lang.ada

Dick Gabriel [the author of the "worse is better" philosophy — su] is welcome to his opinion, but as someone who worked on Multics with the creators of Unix in the 60s I have to say that I don't see it that way. Multics was not rigorously planned first and then executed; our design and implementation evolved over 20 years. We did have this pattern of writing something and discussing it before coding, and often the discussion led to an improved design. Multicians felt that we were rebels, back in the 60s, fighting against batch processing, writing systems in assembler, and using human waves of programmers. We advocated flexibility, virtual memory, interactiveness, and powerful tools. Ken and Dennis did not have the longest hair or wildest beards on the Multics development team.

Multics was written in PL/I. The Bell Labs folks encountered BCPL at Project MAC in the 60s and created B and then C in the 70s. Multics was not "complete" when Bull stopped development on it in the mid 80s: no operating system is ever "complete" since user needs, hardware, and competition keep changing requirements.

For more information, see

http://www.multicians.org/myths.html

http://www.multicians.org/pl1.html

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 6 Jun 2006 08:35:15
Subject: Re: Why C for the Open Source Movement?
Newsgroups: comp.lang.ada

:-) Thanks a lot for correcting me. I am honoured that you gave us this first-hand information. Multics is quite a legend to me, a bit like dragons; now I feel like I've just met an actual dragon slayer :)

Still, would you agree that Multics used The Right Thing approach, as opposed to Unix which uses Worse Is Better?

Object-Oriented Programming and Safety

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 26 Apr 2006 11:17:58
Organization: Adalog
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

> I am working in an Ada based flight software development project. As a means of improving I took to the CASE tool based development. ours is a procedural approach of development,

> but the commercial CASE tools speak a lot about class/object/UML. How will this suit for procedure oriented development where we don't have necessity to identify classes and do detailed design as mentioned in OO approach.

There is no sin in not being object-oriented, it all depends on your needs. Especially in real-time, procedural approaches are often better when ensuring WCET is important.

UML is an object oriented approach, and is not appropriate for procedural development. Other tools supporting SART for example might be more appropriate.

The need must be lead the choice of the tool, not the other way round!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 26 Apr 2006 14:52:50
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

I would not equalize OO with UML.

Ada is a nice language. You'd probably need less tools when working in Ada, either OO or not.

Class/object is a different story. Even if you don't need OOA for your problem space. It depends on how much code you'd like to reuse. You can use generics for that, or tagged types, which are traditionally counted for OO and "classes." But in any case you will need to identify and refactor these.

From: bh <no-spam@nosuchaddress.com>
Newsgroups: comp.lang.ada
Date: Wed, 26 Apr 2006 21:33:04
Subject: Re: procedural vs object oriented

If you don't know how to do good OO with Ada, I'd recommend against it. We tried a project using OO and I think it is pretty safe to say we didn't get what we were hoping for. I think your results will be better with procedural.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 27 Apr 2006 07:22:21
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

According to Robert Dewar during FOSDEM, nobody uses OOP in avionics software, because the uncertainty inherent to dynamic dispatching hinders certification. Is someone on this newsgroup in a position to give a counter-example?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 27 Apr 2006 03:42:10
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

> Can't tell about avionics, but what uncertainty of dynamic dispatching is meant? Or, maybe, "certification" is the context of? Then which certification, according to which criteria?

Dynamic dispatching, by definition, means that you don't know which subprogram you call at run-time. The compiler guarantees that the call will succeed (i.e. that there exists a subprogram to dispatch to), but there is uncertainty about which one it is.

DO-178B does not prohibit dynamic dispatching; it only requires that the program be completely deterministic, and it requires the software developers to provide reasonable proof that the program is indeed deterministic.

If you use dynamic dispatching in a program, you must therefore prove that you know precisely which subprogram you call each time you execute the dispatching call. At DO-178B level A, you must also prove that the machine code in the executable program dispatches correctly and in a deterministic way, in bounded time and memory conditions. This additional burden of proof is on the developer. That's what I meant when I said that dynamic dispatching hinders certification.

The question of "how to I use dynamic dispatching while keeping the certification costs reasonable" is quite interesting, complicated, and has received a lot of thought, but no clear answer has come out of it. So, for now, the only clear-cut answer in the conservative world of avionics is, "you don't."

> Talking about uncertainty in general, what about "inherent uncertainty" of a procedure call? Can you tell which procedures will be called and when at run time? If you can then, you can also do it for dispatching calls. Are generic bodies more certain? With "with function "*" (Left, Right : Foo) return Foo"? Really?

A static procedure call has no uncertainty: when you read the program source, you know exactly which subprogram is called, even in the presence of overloading.

When you instantiate a generic, you also know exactly which subprogram you pass as a parameter. Again there is no inherent uncertainty here.

At Barco, our coding standards prohibit access-to-subprogram values, and require all generics to be preelaborated. Thus they eliminate all uncertainty and make all subprogram calls statically deterministic. Needless to say, our coding standards also prohibit dynamic dispatching.

From: Maciej Sobczak <maciej@msobczak.com>
Organization: CERN — European Laboratory for Particle Physics
Date: Thu, 27 Apr 2006 13:07:35
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

In what way is this better or more certain than a dispatching call based on the tag?:

```
If Shape.Type = Triangle then
  Draw_Triangle(Shape);
elsif Shape.Type = Rectangle then
  Draw_Rectangle(Shape);
else
  Put("Damn, I never thought we will
       have more shape types.");
end If;
```

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 27 Apr 2006 05:03:31
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

Because:

* the dispatching logic is visible in the source and therefore easy to trace to object code (you always certify the machine code, not the source code).

* you see, at the call site, the complete list of possible call targets.

Besides, peer review would reject your code. You should have used a case statement with no "others" clause, and thought about all possible shape types up-front.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: 27 Apr 2006 08:17:25
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

> This is something I'm not getting in this discussion. Provided that you know all derived classes, you know all possible dispatchings and you can validate all for correctness and time the worst one. How's this different than evaluating all branches in a case?
> It's true that you don't have the information at the calling point, but you have it elsewhere.

The issue is not whether or not you *can* validate dispatching calls; of course you can. The issue is *how much it costs* to do so. (This same argument is also why we use Ada instead of C or handwritten assembly).

As you said, the list of all possible call targets is not present at the call site, but spread across the entire program source. You'd have to gather the list of all possible targets for each dispatching call, and review the dispatching machine code *at every call site*. Try to do that on the software you're currently writing. The first step would be for you to come up with a list of all dispatching (not static) calls in your program. If you do just that, you will then start to realise how much effort would be required for full-fledged certification.

Furthermore, during maintenance, more possible call targets can appear, affecting previously tested and certified call sites, and requiring you to redo the certification

process each time you add a type to a derivation class.

In general-purpose programming, you can use all the features of the language that help you write your source code faster; in avionics, you only use those that help you certify your machine code faster. It matters more to reduce the cost of certification than the cost of writing the software, because the former far outweighs the latter.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 27 Apr 2006 22:19:03
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

Ada has unique features designed precisely to help with certification. I suggest you re-read annexes D and H in the light of this thread; you will see why Ada shines where lesser languages fall flat on their face.

Particularly relevant to the present discussion is:

pragma Restrictions (No_Dispatch);

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 27 Apr 2006 16:01:41
Organization: Adalog
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

```
> if Read (File) then
     Foo;
  else
    Bar;
  end if;
```

> The uncertainty of a dispatching call is one of the context, exactly as in the example above. Provided that there is nothing uncertain in how dispatching works or what potential targets do.

Of course you can assume that every dispatching call is equivalent to a case statement over all possibly redefined primitives. That works well for one level.

But if you consider that each called primitive may in turn redispatch internally, you end up with a combinatorial explosion. In theory, yes, the analysis can be performed. In practice, no.

From: Peter Amey <peter.amey@praxis-cs.co.uk>
Date: Thu, 27 Apr 2006 16:38:58
Subject: Re: procedural vs object oriented
Newsgroups: comp.lang.ada

The committee that is revising the DO-178B avionics "standard" is currently meeting in Los Angeles. There is an entire sub-group devoted to the problem of certifying OO software. At the breaks, they look like it is proving hard work!

Those of us in the formal methods sub-group are having a slightly easier time.

From: Ed Falis <falis@verizon.net>
Subject: re: OO vs procedural
Date: Thu, 27 Apr 2006 12:06:27
Newsgroups: comp.lang.ada

There are two papers on AdaCore's website that go into some of the issues with certification of applications containing dispatching:

http://www.adacore.com/2006/03/08/certification-object-orientation-the-new-ada-answer/

http://www.adacore.com/2006/03/30/safety-security-and-object-oriented-programming/

From: kevin cline <kevin.cline@gmail.com>
Date: 4 May 2006 12:40:50
Subject: Re: OO vs procedural
Newsgroups: comp.lang.ada

In the second paper, they give [an example involving an abstract tagged type Alert].

The authors then point out a describe a potential pitfall of this code — that a derived type implementation may fail to call the base implementation. This is true. The authors fail to point out that this possibility could have been prevented by correct base class design.

I also fail to understand why this error is hard to test, but perhaps I do not understand S3 testing methods. I would have expected that a failure of a derived type X_Alert to call the base type Handle method would have been caught by a unit test of X_Alert, when it was observed that after calling X_Alert.Handle, no logging occurred.

I would also expect that the error would be easily detected through any formal verification process, since the erroneous Handle method would not meet the 'Logging occurred' postcondition.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 04 May 2006 22:21:37
Subject: Re: OO vs procedural
Newsgroups: comp.lang.ada

Of course, what you say is true — good unit testing or good peer review will catch the error, and the formal verification process will document how the error was found, corrected, and verified to be corrected. But, by that argument, "any good programmer with a good process can write perfect software in any language, even assembly language".

The point is to help the compiler catch the error automatically, before the first unit test is written and before any peer review takes place. Compile-time checks are why we (in avionics) use Ada in the first place. In other industries, people also like the run-time checks, which help later, i.e. during testing.

Ada and Internationalization

From: Michael Rohan <mrohan@acm.org>
Date: 30 May 2006 16:12:35
Subject: Ada and Internationalization
Newsgroups: comp.lang.ada

I've checked Google and have not been able to find anything in Ada out there for internationalized code. There's support for Wide_Character and Wide_Wide_Character but there doesn't seem to be libraries for message strings. Before starting down the path of writing from scratch, wanted to check.

If nothing is available, I was considering taking Java .properties files, somehow "compiling" them into an Ada package and implementing something akin to Java's MessageFormat:

```
Arguments : Message_Arguments;
...
Arguments.Append ("a string");
Arguments.Append (10);
Arguments.Append (Pi);
Put_Line (Message_Format ("facility",
"msg001", Arguments));
```

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 31 May 2006 07:52:21
Subject: Re: Ada and Internationalization
Newsgroups: comp.lang.ada

GtkAda contains a binding to GNU gettext. Look at http://libre.adacore.com/GtkAda

BTW, with GNU gettext, your internal encoding is likely to be UTF-8, so you wouldn't be using Wide_Character or Wide_Wide_Character.

From: James Dennett <jdennett@cox.net>
Date: Wed, 31 May 2006 07:53:36
Subject: Re: Ada and Internationalization
Newsgroups: comp.lang.ada

> I don't see how this is related to internationalization. It looks like stream communication (see S'Output attribute) or string formatting. In either case you convert data to/from stream/string.

Formatting of strings for human readers needs to produce output that is correctly localized, hence is always an issue in an internationalized program, non?

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Wed, 31 May 2006 12:11:22
Subject: Re: Ada and Internationalization
Newsgroups: comp.lang.ada

When I do this, I make message printing suitable for several languages right from the start, taking advantage of the Ada type system. Collect the messages in an enumeration type, i.e., name them. Then build an (abstract) Message type around this enumeration. Then derive (compose) one type for each natural language.

Advantages:

- Ada's coverage rules will make sure that translation won't miss a single message.

- You can easily create an external messages collection for the translators in XML, Excel, plain text, even gettext if you must, because of the mentioned properties of the message type: You have it in your program, so just write another main unit that basically enumerates the values in the types made for the messages, using the desired output format.

- No need to analyze the entire program using external tools, no need to touch sources.

Disadvantages:

- It's not gettext, only Ada, so maybe it's less fashionable.

- It also requires that a programmer considers messages important enough to be worthy of a type that can be checked by the compiler.

Using a Message type and the .properties approach are somewhat similar, except that with a type, you won't have to leave the Ada language: add the properties to a library package, e.g. in constants, roughly:

```
(en_US => (got_foo => ...,
           no_bar_please => ...,
           argh => ...),
(fr_CA => (got_foo => ...,
           no_bar_please => ...,
           arhg => ...),
...
```

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Organization: cbb software GmbH
Date: Wed, 31 May 2006 17:23:19
Subject: Re: Ada and Internationalization
Newsgroups: comp.lang.ada

I am not sure. It looks like a question of content. Formatting is a quite low level thing. Mixing content and formatting can turn very surprising. There are right-to-left and top-down languages, word ordering might change, their number as well, numerals, ordinals, articles, inflexions etc.

Both as a customer and vendor I always try to avoid internationalized programs. ☺

From: James Dennett <jdennett@cox.net>
Date: Wed, 31 May 2006 08:27:38
Subject: Re: Ada and Internationalization
Newsgroups: comp.lang.ada

And these are included in my notion of internationalized formatting, though for a wide range of languages we can get away with supporting left-to-right, and just dealing with issues of phrase lookup, word ordering and cardinality.

[To Avoid internationalized applications] is increasingly difficult in many domains, though it's certainly true that programming is somewhat simpler when i18n is not a factor.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Organization: cbb software GmbH
Date: Thu, 1 Jun 2006 13:00:07
Subject: Re: Ada and Internationalization
Newsgroups: comp.lang.ada

OK, but even then the target cannot be a simple stupid object like stream or string. It should know how to translate a sequence of "precompiled" objects into a proper sentence. There is a danger that it might quickly become double dispatching, the thing we cannot effectively do in Ada. Alternatively the target object should know the language and be intelligent to determine parts of speech...

So I'd try to stay as much as possible on the side of objects being output. They should know how to translate themselves according to the target locale.

In a recent project I had a similar problem. In place of localization there were different rendering devices: Text, HTML, GTK etc. It ended up with a primitive operation defined on objects, that had a class-wide argument controlling the output parameters. Needless to say, that I am not satisfied with this design.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 2 Jun 2006 20:23:24
Subject: Re: Ada and Internationalization
Newsgroups: comp.lang.ada

I had a similar problem with the formatting tool that produces the Ada Reference Manual and other good stuff. (In fact, it is essentially the same problem.) I ended up defining an "output object" abstract type, with instances of the type for text, HTML, and RTF. The needed operations for formatting and the like are defined for the abstract type. The formatting engine takes an output object and writes to it as needed.

The design was fairly successful; Stephen Leake created a new type and object to support TextInfo output. He was able to do that fairly successfully with the interface already provided.

Ada and XMI

From: marktxx@yahoo.com
Date: 27 May 2006 09:02:35
Subject: Ada to XMI tool?
Newsgroups: comp.lang.ada

Since XMI was discussed recently in this forum. Can anyone name some tools that "reverse engineer" (using this term loosely) Ada 83 code to XMI and/or UML? Commercial or non-commercial.

Static structure only is OK because this would help us identify and reuse existing code in a follow-on design project. (For large projects people still like diagrams.)

Any experience with Rhapsody-Ada, Artisan-Ada or Pragsoft UMLStudio ?

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Sat, 27 May 2006 20:17:49
Subject: Re: Ada to XMI tool?
Newsgroups: comp.lang.ada

Rational Ada ↔ Rational Rose ↔ XMI is the only setup that comes to my mind.

From: Simon Wright <simon@pushface.org>
Date: Sun, 28 May 2006 10:21:02
Subject: Re: Ada to XMI tool?
Newsgroups: comp.lang.ada

No experience, but I know that Artisan will reverse-engineer Ada 95 (to Artisan's UML Ada-profile) so you stand a reasonable chance.

From: pth@darrach.net
Date: 29 May 2006 02:33:29
Subject: Re: Ada to XMI tool?
Newsgroups: comp.lang.ada

You can try Headway Review, http://headwaysoftware.com/products/review/. Review uses advanced reverse engineering and static analysis techniques to create a powerful code comprehension, code review, and source code visualization tool for Architects and Team Leads.

It will certainly help you understand what parts of the code you want to reuse.

One caveat: you don't specify what compiler you use for the Ada 83 code. If the code compiles with GNAT then its a no-brainer, you get the model instantly from the GNAT adt files, no need to spend time laying out diagrams or struggling with the reverse engineering. But I should also point out that we have a number of customers using Review with non-GNAT based code, I know for certain it includes Greenhills and SunAda. If you download and try it out, one of our guys will be happy to help you get up and running.

ASIS to XML Schema

From: Simon Wright <simon@pushface.org>
Date: Tue, 23 May 2006 22:01:47
Subject: Re: ANN: Ada source code decorator
Newsgroups: comp.lang.ada

Georg Bauhaus wrote:

> [...] using Simon Wright's ASIS based GIs.

Although that work was useful, a lot more needs to be done on it to make it anywhere near complete. It would be a lot easier if ASIS (the standard) was open-source.

If there is to be an ASIS 2005 I hope the representation will be as an XML schema and not an API; the API makes using ASIS so very clumsy compared to the various XML processing technologies available now.

Well, that's the way it seems from here!

From: Stephen Leake <stephen_leake@acm.org>
Date: Wed, 24 May 2006 20:28:17
Subject: Re: ANN: Ada source code decorator
Newsgroups: comp.lang.ada

The ASIS _standard_ is open-source; it consists of Ada specs giving the API.

The GNAT implementation is also open-source. So what is missing?

There [will be an ASIS 2005]; the ARG is starting work on it.

I don't see how [using XML Schema instead of an API] is remotely possible. The whole point of ASIS is to access the knowledge the compiler has about the source. Thus the compiler has to provide functions to access that knowledge.

I can see building an XML processor in which some of the XML tags cause ASIS API calls. But that is a layer on top of the ASIS API, not a replacement for the ASIS API.

Can you give a concrete example of how an ASIS-XML would work?

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 25 May 2006 14:36:12
Subject: Re: ANN: Ada source code decorator
Newsgroups: comp.lang.ada

The compiler could provide a representation of its knowledge of the source that uses XML. For example, internal trees (or graphs), where the nodes carry information not only about names, etc, but also about representations. Trees, or groves, are an ideal candidate for XML based representation because an XML document instance is a tree or graph, and XML based tools are made for working on these.

In fact, ASIS and XML use almost the same wording, e.g. element trees.

If ASIS were to provide a set of XML element definitions, these could be used to write transformations to and from XMI, easing the use of Ada with some OOD tools.

Getting a sorted index of all tagged types in a program will then be very easy with XML tools: just declare the corresponding XPath choice and employ XSL's sort and key functions. Done.

Which leads me to a question: Is there a set of "ASIS items" that you use frequently? Items that could be useful when represented using XML, even when the Ada information contained therein is limited?

From: Stephen Leake <stephen_leake@acm.org>
Date: Thu, 25 May 2006 09:48:13
Subject: Re: ANN: Ada source code decorator
Newsgroups: comp.lang.ada

Ok, now I see; that does make sense.

I don't think that will happen as part of the ASIS 05 effort; there are not many people working on it, and they don't see that as their mission. They are working on extending the ASIS 95 API to cover the new Ada 2005 features, and possibly fixing any problems with the current API.

I believe the original decision to specify an API, rather than a format, was to allow each compiler vendor to use a format that was similar to their already-existing internal compiler structures, thus making it easier for each compiler vendor to support ASIS.

I suspect that argument still holds. But it might be reasonable to add an optional standard XML format.

If enough people want it to happen, they should get together and produce a working example of such a standard format, then lobby to get it approved as a secondary standard.

Note that vendors will only support such a format if they see money in it. So people who want it must be able to say they will pay a reasonable fee for it, and that there are still others who will also pay such a fee. Or they need to fund the entire development themselves.

I have not used XML processors, so I can't comment on whether they are "less clunky" than using the ASIS API. But I can see that having a standard XML representation of the Ada source would make it easier for people who are familiar with XML to build useful tools. And since there are certainly more people familiar with XML than with ASIS, that would be a good thing for Ada.

Ada vs. Fortran

From: Nasser Abbasi <nma@12000.org>
Date: Mon, 22 May 2006 04:54:42
Subject: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

I like to discuss the technical reasons why Ada is not used as much as Fortran for scientific and number crunching type applications?

To make the discussion more focused, let's assume you want to start developing a large scientific application in the domain where Fortran is commonly used.

Say you want to develop a new large Finite Elements Methods program or large computational physics simulation system. Assume you can choose either Ada or Fortran.

What are the technical language specific reasons why Fortran would be selected over Ada?

I happened to know a little about Ada and Fortran, and from what I know, I think Ada would be an excellent choice due to its strong typing, good support for numerical types and good Math library.

I know also that Fortran is supposed to be better/faster when it comes to working with large Arrays (Matrices), but it is not clear to me why that is, and if it is still true with Ada 05. Something about arrays aliasing, but not sure how that is.

I am also not sure on the support of sparse matrices in both languages' libraries.

It is known that Ada strong domain is realtime and safety critical applications. I never understood why Ada never became popular in the scientific field in particular in areas such as computational physics or CFD or such similar fields.

From: Dan Nagle <dannagle@verizon.net>
Date: Mon, 22 May 2006 15:23:05
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

> Some immediate reasons:
> 1) Packaging. Packages allow better organization of software, which is good for any kind of application.

Can you compare and contrast Ada packages with Fortran modules and submodules?

> 2) Strong typing. Scientific applications often deal with physical units, and Ada is great at supporting these.

What specific features of Ada provide better support than the comparable feature of Fortran?

> 3) User defined accuracy. Ada allows you to define the accuracy you need, the compiler chooses the appropriate representation. Note that you are not limited to only two floating point types (many machines have more than that).

How is this better than Fortran's kind mechanism?

> 4) Fixed points. Not available in Fortran

Agreed. How important is this for floating point work? Fortran is rarely used for imbedded software (at least, I wouldn't).

> 5) Guaranteed accuracy, not only for basic arithmetic, but for the whole mathematical library

Can you compare Ada's accuracy requirements with Fortran's support for IEEE 754?

> 6) Standardization. All compilers process exactly the same language.

Again, how is this different? Fortran compilers are required to be able to report use of extensions to the standard.

> 7) Interfacing. Easy to call libraries in foreing languages => all libraries available for Fortran are available for Ada.

Can you compare Interfaces.C to ISO_C_BINDING? How is one better or worse than the other?

> 8) Concurrency, built into the language

Co-arrays and concurrent loops are coming in Fortran 2008.

> 9) Generics. Stop rewriting these damn sorting routines 1000 times.

Intelligent Macros are coming in Fortran 2008.

> 10) Default parameters. Makes complex subprograms (simplex...) much easier to use.

Agreed.

> 11) Operators on any types, including arrays. Define a matrix product as "*"...

How is Ada's operators for types better or worse than Fortran's? Is Ada's "*" operator better than Fortran's matmul()?

> 12) Bounds checking, with a very low penalty. Makes bounds checking really usable.

How is Ada's bounds checking better or worse than Fortran's?

"Fortran" /= "FORTRAN 77" ;-)

From: Jean-Pierre Rosen <rosen@adalog.fr>
Organization: Adalog
Date: Tue, 23 May 2006 10:25:37
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

[...] Because Fortran has no fixed points, the scientific community sees floating point as the only way to model real numbers.

Actually, fixed points have nothing to do with embedded software, they are a different way of modelling real (in the mathematical sense) numbers, with different numerical properties. Depending on the problem, fixed point may (or not) be more appropriate.

[...] Ada's accuracy requirement is independent from any hardware (or software) implementation of floating points, and are applicable even for non IEEE machines.

[...] Concurrency has been in Ada since 1983! Moreover, it's a multi-tasking model, not concurrent statements model. Both models have benefits and drawbacks, it depends on the needs.

[...] I don't know what an "intelligent macro" is, but certainly generics (once again available since 1983"), are much more than macros, even intelligent ones.

For one thing, the legality of generics is checked when the generic is compiled. This means that, provided actual parameters meet the requirements of the formals, there is no neeed to recheck at instantiation time, and ensures that any legal instantiation will work as expected. AFAIK, this cannot be achieved by macros.

[...] I may miss something on the Fortran side, but Ada's very precise typing allows to define variables whose bounds are delimited.

If these variables are later used to index an array (and if the language features are properly used), the compiler statically knows that no out-of-bound can occur. In short, most of the time, an Ada compiler is able to prove that bounds checking is not necessary, and corresponding checks are not generated.

In practice, compiling an Ada program with or without bounds checking shows very little difference in execution speed, because only the really useful checks are left, all the spurious ones have been eliminated.

From: "J.F. Cornwall" <JCornwall@cox.net>
Organization: U.S. Geological Survey, Reston VA
Date: Wed, 24 May 2006 12:56:29
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

> In a weather forecasting program you want to have data acquisition (real-time), prediction (computation) and display (real-time GUIs) running on a continuous, high uptime basis across a network of machines.

If Fortran had strong multitasking, real-time and distributed capabilities, these goals would be reasonable and achievable within the language. Absence of these features means such systems would often (I guess) be multi-language setups, with things like Java, C++, Tcl/Tk, shell scripts, cron jobs etc. playing a part.

Has anyone here worked on a big meteorological system? Am I right?

In my US Air Force days, I worked at a large global weather-forecasting facility. We had multiple data input systems (a variety of comm links talking to several Univac mainframes), multiple number-crunching systems (a couple more Univacs and a Cray), and an cluster of 40 or so Vax 11/780s for interactive tweaking of the forecasts. The majority of the software for the comm was in assembler, just about all of the remainder was Fortran (IV and 77, this was back in the early 80's...).

We also used Fortran mixed with assembly code for a new comm front-end machine that was implemented in '88. Fortran was used for comm, utility programs, forecasting models, database input/output/maintenance, and just about everything else in that system. Worked fine.

Nowadays, I have no idea what they're running. Bet there's still a lot of Fortran though :-)

From: "Dr. Adrian Wrigley" <amtw@linuxchip.demon.co.uk.uk.uk>
Date: Wed, 24 May 2006 13:39:44
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

Interesting.

I *think* you are supporting my view that in practice, Fortran requires additional support or coding outside of the language to tie together the different parts of a complex system.

You speak of utility programs, forecasting programs, database I/O programs. Invoking these in the right order, at the right time, on the right files at the right terminals is always done outside of the pure Fortran application. At the very least it requires an OS command interpreter. It probably involves scripts to delete old files or do other housekeeping.

In Ada, the separate program components can form a *single* running application program entity, with a single invocation - even if the program is running across several loosely connected machines and consists of many different executable files. The program execution is a network of cooperating processes and shared data stores. Parts of the program can even be recompiled as it runs - without affecting the shared data stores or other executing tasks.

In fact Ada supports persistent variables with hold their values even if the program is stopped completely and restarted later. No mainstream language comes even close to this program execution model.

From: "J.F. Cornwall" <JCornwall@cox.net>
Organization: U.S. Geological Survey, Reston VA
Date: Wed, 24 May 2006 16:49:10
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

Actually, in that particular environment, everything was tied together in a complicated web of cross-ties. The Fortran code couldn't do everything, the assembly code couldn't do everything, the scripting and batch control languages couldn't do everything. etc... That would have been the case had we been using Ada, as well.

And we did look at Ada when starting out on the comm front-end project. At that time it wouldn't do what we needed it to do, so we went with a continuing mixture of F77 and assembler. Sorry, I don't recall the details of what we needed that it couldn't do, recall that this was in the early 1980s.

From: "Dr. Adrian Wrigley" <amtw@linuxchip.demon.co.uk.uk.uk>
Date: Wed, 24 May 2006 18:08:20
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

You're quite right.

But one thing apparent in this discussion is that the Ada programmer's view of Fortran is the FORTRAN 77 many learned in college, but the Fortran programmer's view of Ada is of Ada 83, when that was hot technology. Neither view has much relevance in determining the technical suitability of the contemporary languages for new projects.

I'd be overselling the features of modern Ada to say that the scripting and batch control 'glue' can *all* be done within the language - but a huge part of it can be. And this brings a major benefit to system portability, complexity and integrity.

From: Dick Hendrickson <dick.hendrickson@att.net>
Subject: Re: Ada vs Fortran for scientific applications
Date: Wed, 24 May 2006 17:12:55
Newsgroups: comp.lang.ada,comp.lang.fortran

Dr. Adrian Wrigley wrote:

> So what does the standard say must happen if you attempt such an access? Can a program fail unpredictably under such (rather common!) circumstances - as routinely happens in C and C++, sometimes at great cost?

The Fortran standard says nothing at all about what must happen for most run-time errors. There is a requirement that a compiler be able to diagnose syntax-like errors at compile time. There is also a requirement that some (unspecified) I/O errors and some memory management errors be checked for at run time. The job will abort unless the program uses one of the error detection methods. But for things like subscript bounds errors, or subroutine argument mismatches, the standard doesn't impose anything on the compiler.

In general, the standard imposes restrictions on standard conforming programs, not on the compiler. This allows compilers to extend the standard in "useful" ways. Technically, a standard conforming program is not allowed to use these extensions, but many do ;). Most compilers implement a command line option to do enhanced syntax checking and report use of extensions.

Subscript bounds errors usually go unchecked and do whatever they do. They're really fun to debug because adding a PRINT statement usually moves the effect to some other part of the program. This isn't Fortran's greatest strength ☺ It was a compromise between safety and speed.

The other big problem with (old) Fortran programs was messing up the argument list in a procedure call. Separate compilation made this a lot easier to do. The Fortran 90 addition of MODULES essenially closes this hole. Most procedure interfaces now can be explicit and the compiler must check for calling consistency.

It's harder to shoot yourself in the foot now, but people can still lie to the compiler.

From: "Dr. Adrian Wrigley" <amtw@linuxchip.demon.co.uk.uk.uk>
Subject: Re: Ada vs Fortran for scientific applications
Date: Wed, 24 May 2006 17:54:12
Newsgroups: comp.lang.ada,comp.lang.fortran

I think this is an area that Ada really shines. The standard requires numerous checks for consistency at both compile time and runtime. Versions of code that don't match properly can't be linked together or can't be run together (as appropriate). Using the language gives a feeling of integrity of coding, with mistakes often being caught very early on.

Unfortunately, the language features for integrity cannot be added to an existing language without breaking old code. This is because the integrity features are often a result of prohibiting "dodgy" code, flawed syntax or misfeatures.

The history of the C family of languages illustrates this. I'm not sure where modern Fortran sits in relation to its forbears in terms of safety and security though.

It's noteworthy that Ada and Fortran are on convergent paths (modules, user defined types, templates etc).

With array subscripts, an exception must be raised if the bounds are exceeded. The same with arithmetic operations. Curiously, compiling Ada under gcc (GNAT), a compilation switch is needed to be standards compliant - a mistake:(). The checks can be switched on and off in the source code as desired.

One of the benefits of the compile- and run-time checking is that refactoring code becomes much easier because the compiler will usually tell you about what parts haven't been fixed up yet.

Languages like C or Perl are at the opposite end of the spectrum, I find. From what I read here, Fortran is somewhere in between.

From: Gordon Sande <g.sande@worldnet.att.net>
Subject: Re: Ada vs Fortran for scientific applications
Date: Wed, 24 May 2006 18:34:58
Newsgroups: comp.lang.ada,comp.lang.fortran

There is a distinction to be made between what the standard requires and what the various compilers offer. Some systems are oriented to the ultimate SpecMark(??) benchmark figures while others offer tightly monitored executions.

Subscript checking can be turned on for those systems. Some even go the extra mile of offering checking for usage of undefined (uninitialized) variables. Some undefineds can be caught as a byproduct of flow checking at compile time but others, like array elements, are only possible at run time.

Some "real" programmers disdain the use of such tools but others are glad for all the aids that are available.

As with most groups there are subgroups. Some Fortran programmers dismiss any notions of less than full exploitation of every last quirk of the hardware and software of the day. Their equivalents in other programming groups are probably the folks who ignore all interrupts.

The urban legends have the Fortran error of a DO loop that changed into an assignment because of a typo changing a comma into a period and a satellite was lost. For Ada it is a tossed interrupt that caused a launch failure. Bad practice of one will always be inferior to good practice of the other.

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Wed, 24 May 2006 21:36:29
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

> Well, at least one thing is common between Ada and Fortran: Both are case INSENSITIVE.

Two other points in common are readability (or, non-cryptic syntax) and (Fortran: 77+ ?) full-bracketing (conditional or loop statements terminated by END). Both things are extremely helpful for revising code, which is crucial for scientific programming, and separate the pre- (Pascal, C) and post-1977 compiled languages.

From: Richard E Maine <richard.maine@nasa.gov>
Subject: Re: Ada vs Fortran for scientific applications
Date: Wed, 24 May 2006 12:56:51
Organization: NASA Dryden
Newsgroups: comp.lang.ada,comp.lang.fortran

I had also noticed the similarity between Fortran 90 modules and Ada packages. Not idenical by any means, but there are sure some similarities.

And the possibility of specifying procedure arguments by keyword instead of just positionally. You find that in some scripting languages. And you find things like that in lots of other contexts,

including the syntax typically used to invoke compilers. But in compiled languages, it seems like the feature is rare; it is shared by Fortran 90 and Ada, and then I start slowing down a lot in naming compiled languaages in widespread use that have it.

From: "Ed Falis" <falis@verizon.net>
Date: Wed, 24 May 2006 18:40:34
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

I have to say as an Ada guy, that I'm finding this thread more interesting than most language comparison fests. You Fortran guys are presenting mature, intelligent and interesting perspectives. Kudos to you.

From: Brooks Moses <bmoses-nospam@cits1.stanford.edu>
Organization: Stanford University
Date: Thu, 25 May 2006 15:31:42 -0700
Subject: Re: Ada vs Fortran for scientific applications
Newsgroups: comp.lang.ada,comp.lang.fortran

And kudos to you as well — I had just been thinking much the same thing about the Ada crossover. I've found it a very thought-provoking thread!

Conference Calendar

This is a list of European and large, worldwide events that may be of interest to the Ada community. Further information on items marked ♦ is available in the Forthcoming Events section of the Journal. Items in larger font denote events with specific Ada focus. Items marked with ☺ denote events with close relation to Ada.
The information in this section is extracted from the on-line *Conference announcements for the international Ada community* at: http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programs, URLs, etc. and are updated regularly.

2006

☺ July 02 — 5th **International Workshop on Constructive Methods for Parallel Programming** (CMPP'2006), Kuressaare, Estonia. Topics include: formal models, methods, and languages for parallel programming; parallelization and compilation techniques; parallel and distributed object-oriented programming; hardware-software codesign; etc.

☺ July 03-07 — 20th **European Conference on Object-Oriented Programming** (ECOOP'2006), Nantes, France. Topics include: Patterns, Modularity, Adaptability, Separation of Concerns, Components, Frameworks, Concurrency, Real-time, Embedded, Distribution, Domain Specific Languages, Language Workbenches, Multi-paradigm Languages, Language Innovations, Compilation, Methodology, Practices, Metrics, Formal methods, Tools, etc.

- ☺ July 03 — **Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems** (ICOOOLPS'2006). Topics include: implementation of fundamental OOL features: inheritance (object layout, late binding, subtype test, ...), genericity (parametric types), memory management; runtime systems: compilers, linkers, etc; optimizations: static and dynamic analyses, etc; resource constraints: real-time systems, etc: relevant choices and tradeoffs: separate compilation vs. global compilation, dynamic checking vs. proof-carrying code, etc.
- ☺ July 03 — 10th **Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts**. Topics include: experiences, ideas and resources to support the teaching and learning of basic object-oriented concepts.
- ☺ July 03 — 6th **Workshop on Parallel/High-Performance Object-Oriented Scientific Computing** (POOSC'2006). Topics include: tried or proposed programming language alternatives to C++; issues specific to handling or abstracting parallelism; etc.
- July 03 — 1st **Workshop on Domain-Specific Program Development** (DSPD'2006). Topics include: Role of language paradigms (e.g., object-oriented) in domain-centric software development processes; Tools to support domain-specific modeling transformation and domain-specific language implementation; Relationship between domain-specific modeling and domain-specific languages; Metrics, benchmarks, techniques and tools to assess the benefits of domain-specific modeling and languages (e.g., productivity, reliability, robustness, maintenance and evolution of software components); Relationship between domain-specific languages and scripting languages, general-purpose languages, markup languages, etc.
- ☺ July 04 — 3rd **International Workshop on Practical Problems of Programming in the Large** (PPPL'2006) Theme: "Industrial Problems, Technology Transfer, Research Validation". Topics include: Experience, positive or negative with technology transfer and cooperation of academia and industry; Negative results: what went wrong although it should have worked according to software engineering folklore; Success-stories for component-based software engineering; Keeping systems with large amounts of classes / objects / modules / components organised; Refactoring, Software Evolution and Migration; etc.

July 04-07 — 26th **International Conference on Distributed Computing Systems** (ICDCS'2006), Lisboa, Portugal. Topics include: all aspects of distributed and parallel computing.

☺July 05-07 18th **Euromicro Conference on Real-Time Systems** (ECRTS'2006), Dresden, Germany. Topics include: all aspects of real-time systems; special focus on industrial applications of real-time technology; compiler support; component-based approaches; middleware and distribution technologies; programming languages; real-time operating systems; model-driven development of embedded RT systems; formal methods; reliability, security and survivability in RT systems; scheduling and schedulability analysis; worst-case execution time analysis; validation techniques; etc.

July 09-16 33rd **International Colloquium on Automata, Languages and Programming** (ICALP'2006), Venice, Italy. Topics include: Principles of Programming Languages, Formal Methods, Models of Concurrent and Distributed Systems, Program Analysis and Transformation, etc.

☺July 10-13 **OMG Workshop on Distributed Object Computing for Real-time and Embedded Systems**, Washington, DC, USA. Topics include: Real-time systems; Embedded systems; Fault-tolerant systems; High-availability systems; Safety-critical systems; Design tools for real-time distributed systems; Real-time middleware, including real-time CORBA; Modeling notations, including UML; Model-Driven approaches, including MDA; High-level real-time programming models; etc.

July 10-14 2nd **European Conference on Model Driven Architecture: Foundations and Applications** (ECMDA-FA'2006), Bilbao, Spain. Topics include: Model Transformation - languages, tools; MDA for Large Scale Distributed Systems; Comparative studies of MDA tools; MDA for Legacy Systems; MDA for systems engineering; MDA for embedded systems; MDA for high-integrity systems (safety-critical and security-critical systems; etc.

☺July 12-15 12th **International Conference on Parallel and Distributed Systems** (ICPADS'2006), Minneapolis, Minnesota, USA. Topics include: Parallel and Distributed Applications and Algorithms; Reliable and Fault-Tolerant Computing; Real-Time Systems; Tools, and Evaluation; etc.

July 17-21 Absolute Software - Public Ada 95 Course, Carlsbad, CA, USA

☺July 23-26 25th **Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing** (PODC'2006), Denver, Colorado, USA. Topics include: Concurrent programming, Distributed systems and middleware platforms, Correctness and verification of distributed and parallel programming, etc.

☺August 14-18 35th **International Conference on Parallel Processing** (ICPP'2006), Columbus, Ohio, USA. Topics include: findings in any aspects of parallel and distributed computing; such as Compilers and Languages, Systems Support for Parallel and Distributed Applications, etc.

August 21-27 14th **International Symposium of Formal Methods Europe** (FM'2006), Hamilton, Canada. Topics include: Tools for formal methods (tool support and software engineering, environments for formal methods), Formal methods in practice (experience with introducing formal methods in industry, case studies), etc.

☺August 26-27 11th **International Workshop on Formal Methods for Industrial Critical Systems** (FMICS'2006), Bonn, Germany

☺Aug 29 – Sep 01 12th **International Conference on Parallel and Distributed Computing** (Euro-Par'2006), Dresden, Germany. Topics include: the promotion and advancement of parallel computing; Support Tools and Environments; Distributed Systems and Algorithms; Parallel Programming: Models, Methods, and Languages; Embedded Parallel Systems; etc.

☺September 13-15 7th **Joint Modular Languages Conference** (JMLC'2006), Oxford, England. Topics include: programming language design and implementation; software tools and environments; software quality and testing; formal methods in modular and composable software development; modularity and composability in parallel and distributed systems; modularity and composability in safety-critical and real-time systems; software engineering education; case studies aligning with any of the above; etc.

September 13-15 3rd **International Workshop on Rapid Integration of Software Engineering techniques** (RISE'2006), Geneva, Switzerland. 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, etc. Key applications domains: High-availability or mission-critical systems, Embedded systems and applications, Development environments, etc. Deadline for registration: July 3, 2006

September 16-17 2nd **International Workshop on Views On Designing Complex Architectures** (VODCA'2006), Bertinoro, Italy. Topics include: all areas related to the design of complex architectures; such as Formal methods for security, Language-based security, Availability properties, Component-based design, Distributed systems, etc.

☺ September 16-20 **Parallel Computing Technologies** (PaCT'2006), Seattle, Washington. USA. Topics include: Compilers and tools for parallel computer systems, Parallel programming languages and applications, Run time system support for parallel systems, Parallel processing in type safe languages, Support for correctnes in hardware and software (esp. with concurrency), etc.

September 17-20 7th **Conference on Communicating Process Architectures** (CPA'2006), Edinburgh, Scotland, UK. Topics include: all aspects of concurrency - theory and practice, software/middleware/hardware, and applications.

September 18-19 6th **International Workshop on Automated Verification of Critical Systems** (AVoCS'2006), Nancy, France. Topics include: tools and techniques for the verification of critical systems; such as automated verification, including model checking, theorem proving, abstract interpretation, and refinement pertaining to various types of critical systems (safety-critical, security-critical, business-critical, performance-critical, ...). Deadline for submissions: July 28, 2006 (short presentation abstracts)

☺ September 18-20 20th **International Symposium on DIStributed Computing** (DISC'2006), Stockholm, Sweden. Topics include: concurrent programming and synchronization algorithms; fault tolerance; specification, semantics, and verification; distributed programming languages; distributed object-oriented computing; etc.

☺ September 20-22 *Real-Time and Networked Embedded Systems Track* of the 11th **IEEE International Conference on Emerging Technologies and Factory Automation** (RTNES-EFTA'2006), Prague, Czech Republic. Topics include: Real-time (distributed) embedded systems; Dependable embedded systems; Formal methods; Real-time executives and operating systems; Real-time scheduling; Real-time components and Middleware; Software engineering and programming languages; Case studies (industrial automation, automotive, avionics, communications...); etc.

☺ September 20-22 19th **International Conference on Parallel and Distributed Computing Systems** (PDCS'2006), San Francisco, California, USA. Topics include: all areas of Parallel and Distributed Computing Systems, their modeling and simulation, design, use and performance, and their impact; such as on Languages, Compilers and Operating Systems; Libraries and Programming Environments; Software Development, Services, Support, and Tools; Middleware for Parallel and Distributed Computing; Embedded Systems; Parallel and Distributed Applications; etc.

☺ September 21-23 6th **Austrian-Hungarian Workshop on Distributed and Parallel Systems** (DAPSYS'2006), Innsbruck, Austria. Topics include: Parallel and distributed programming languages and algorithms, Formal models for parallel and distributed computing, Software engineering and development tools, etc.

September 24-27 22nd **IEEE International Conference on Software Maintenance** (ICSM'2006), Philadelphia, PA, USA. Topics include: maintaining, modifying, enhancing, and testing operational systems, and designing, building, testing, and evolving maintainable systems.

September 25-28 26th **IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems** (FORTE'2006), Paris, France. Special focus on verified middleware and distributed services. Topics include: Practical experience with formal methods, etc.

September 27-29 9th **International Conference on Quality Engineering for Software-Based Systems** (CONQUEST'2006), Berlin, Germany. Topics include: first-hand information on the practical use and further development of methods and techniques; specific real-life case studies with detailed quality analysis and evaluation; capabilities and availability of quality engineering tools, etc.

September 27-29 6th **IEEE International Workshop on Source Code Analysis and Manipulation** (SCAM'2006), Philadelphia, PA, USA. Topics include: program transformation, abstract interpretation, program slicing, source level software metrics, program comprehension, etc.

October 01-06 9th **International Conference on Model-Driven Engineering Languages and Systems** (MoDELS'2006), Genoa, Italy. Topics include: Model-driven engineering methodologies, approaches, languages and tools; Domain-specific modeling languages; Programming language and

metaprogramming support for linking models to code; Modeling languages and tools; Semantics of modeling languages; Modeling and analysis of real-time, embedded, and distributed systems; etc.

☺ October 02-04 — 25th **IEEE International Symposium on Reliable Distributed Systems** (SRDS'2006), Leeds, UK. Topics include: reliability, availability, safety, security, and real time; Security and high-confidence systems, Distributed objects and middleware systems, Formal methods and foundations for dependable distributed computing, Analytical or experimental evaluations of dependable distributed systems, etc.

☺ October 12-13 — **Automotive - Safety & Security 2006**, Stuttgart, Germany. Theme: "Sicherheit und Zuverlässigkeit für automobile Informationstechnik". Organized by Gesellschaft für Informatik (GI), etc., in cooperation with Ada-Deutschland and Fachgruppe "Ada", etc. Topics include (in German): Zuverlässigkeit und Sicherheit für fahrbetriebs-kritische Software und IT-Systeme; Sichere Entwicklung, Aktualisierung und Freischaltung; Normen und Standardisierungsbestrebungen; Entwicklungsbegleitende Evaluation und Zertifizierung; etc.

☺ October 18-20 — **IEEE Symposium on Industrial Embedded Systems** (IES'2006), Antibes, Juan les Pins, Cote d'Azur, France. Topics include: recent developments, deployments, technology trends and research results, as well as initiatives related to embedded systems and their applications in a variety of industrial environments. Deadline for submissions: July 15, 2006 (work-in-progress)

☺ October 22-26 — 21st **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2006), Portland, Oregon, USA. Topics include: diverse disciplines related to object technology. Deadline for submissions: August 1, 2006 (Student Volunteers)

October 22-26 — 5th **International Conference on Generative Programming and Component Engineering** (GPCE'2006), Portland, Oregon, USA. Co-located with OOPSLA'2006. Topics include: Generative techniques for Product-line architectures; Distributed, real-time and embedded systems; Model-driven development and architecture; Component-based software engineering (Reuse, distributed platforms and middleware, distributed systems, evolution, patterns, development methods, deployment and configuration techniques, and formal methods); Integration of generative and component-based approaches; Industrial applications; etc.

October 23-26 — 4th **International Symposium on Automated Technology for Verification and Analysis** (ATVA'2006), Beijing, China. Topics include: theory useful for providing designers with automated support for obtaining correct software or hardware systems, applications of theory in engineering methods and particular domains and handling of practical problems occurring in tools, etc.

October 23-27 — 13th **Working Conference on Reverse Engineering** (WCRE'2006), Benevento, Italy. Theme: "Empirically Assessing Reverse Engineering Techniques and Tools". Topics include: Empirical studies in reverse engineering; Decompilation and binary translation; Redocumenting legacy systems; Reverse engineering tool support; Mining software repositories; Program analysis and slicing; Software architecture recovery; Program transformation and refactoring; etc.

October 25-27 — 5th **International Conference on Software Methodologies Tools, and Techniques** (SoMeT'2006), Quebec, Canada. Topics include: Software methodologies, and tools for robust, reliable, non-fragile software design; Automatic software generation versus reuse, and legacy systems, source code analysis and manipulation; Software evolution techniques; Formal methods for software design; Static and dynamic analysis, and software maintenance; Formal techniques for software representation, software testing and validation; Software reliability, and software diagnosis systems; etc.

October 26-28 — 6th **International Conference on Quality Software** (QSIC'2006), Beijing, China. Topics include: Software quality (reliability, safety and security, ...); Methods and tools; Evaluation of software products and components (static and dynamic analysis, validation and verification); Formal methods (program analysis, model checking, formal process models, ...); Applications (component-based systems, distributed systems, embedded systems, enterprise applications, safety critical systems, ...); etc.

☺ Oct 29 – Nov 03 — 8th **International Symposium on Distributed Objects and Applications** (DOA'2006), Montpellier, France. Topics include: Application case studies of distribution technologies; Component-based software development; Design patterns for distributed systems; Integrated development environments; Middleware for distributed object computing; Real-time solutions for distributed objects; Technologies for reliability and fault-tolerance; Testing and validation of distributed object systems; etc.

Oct 30 – Nov 03 8th **International Conference on Formal Engineering Methods** (ICFEM'2006), Macao SAR, China. Topics include: Abstraction and refinement; Tool development and integration for formal system design, analysis and verification; Integration of formal verification tools in CASE tools; Techniques for specification, verification and validation; Techniques and case studies for correctness by construction; Experiments of verified systems; Application in real-time, hybrid and critical systems; Emerging technologies; etc.

November 08-10 4th **Asian Symposium on Programming Languages and Systems** (APLAS'2006), Sydney, Australia. Topics include: both foundational and practical issues in programming languages and systems; type systems, language design; program analysis, optimization; software security, safety, verification; compiler systems, interpreters; programming tools and environments; etc.

♦Nov 12-16 **2006 ACM SIGAda Annual International Conference** (SIGAda'2006), Albuquerque, New Mexico, USA. Sponsored by ACM SIGAda, in cooperation with SIGAPP, SIGCAS, SIGCSE, SIGPLAN, SIGSOFT, Ada-Europe, and Ada Resource Association (ACM approval pending, Cooperation approvals pending.) Topics include: reliability needs and styles; safety and high integrity issues; analysis, testing, and validation; standards; use of ASIS for new Ada tool development; mixed-language development; Ada in XML and .NET environments; quality assurance; Ada & software engineering education; commercial Ada applications: what Ada means to the bottom line; static and dynamic code analysis; software architecture and design; etc.

☺December 01-04 4th **International Symposium on Parallel and Distributed Processing and Applications** (ISPA'2006), Sorrento, Italy. Topics include: Parallel/distributed system architectures; Tools and environments for software development; Parallel/distributed algorithms; Distributed systems and applications; Reliability, fault tolerance, and security; etc. Includes "Languages and Algorithms" and "Software and Applications" Tracks.

☺December 04-07 7th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2006), Taipei, Taiwan. Topics include: Parallel/distributed architectures; Reliability, and fault tolerance; Formal methods and programming languages; Parallelizing compilers; Component-based and OO Technology; Tools and environments for software development; Parallel/distributed algorithms; Task mapping and job scheduling; etc.

December 05-07 19th **International Conference on Software & Systems Engineering and their Applications** (ICSSEA'2006), Paris, France. Topics include: distributed systems, real-time systems, embedded systems, interoperability, evolution, object-orientation, formal methods, validation, certification, reliability, etc.

☺December 05-08 27th **IEEE Real-Time Systems Symposium** (RTSS'2006), Rio de Janeiro, Brazil. Topics include: all aspects of real-time systems design, analysis, implementation, evaluation, and case-studies.

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

2007

January 03-06 *Software Technology Track* of the 40th **Hawaii International Conference on System Sciences** (HICSS-40), Waikoloa, Big Island, Hawaii, USA. Includes mini-tracks on: Software Engineering Decision Support (topics include: Design decisions; Reuse decisions; Maintenance decisions; Selection of software tool, methods or techniques; ...); etc.

January 15-16 **ACM SIGPLAN 2007 Symposium on Partial Evaluation and Program Manipulation** (PEPM'2007), Nice, France. Co-located with POPL'2007. Topics include: program manipulation, partial evaluation, and program generation. PEPM focuses on techniques, theory, tools, and applications of analysis and manipulation of programs. Deadline for submissions: October 18, 2006 (abstracts), October 20, 2006 (papers)

January 17-19 34th **Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages** (POPL'2007), Nice, France. Topics include: fundamental principles and important innovations in the design, definition, analysis, transformation, implementation and verification of programming languages, programming systems, and programming abstractions. Deadline for submissions: July 15, 2006

☺February 07-09 15th **Euromicro Conference on Parallel, Distributed and Network-based Processing** (PDP'2006), Naples, Italy. Topics include: Advanced Applications (scientific and engineering applications, multi-disciplinary and multi-component applications, real-time applications, ...); Models and Tools for Programming Environments; Distributed Systems; Languages, Compilers and Runtime Support Systems (task and data parallel languages, object-oriented languages, dependability issues, ...); Parallel Computer Systems

Mar 24 – Apr 01 13th **International Conference on Tools and Algorithms for the Construction and Analysis of Systems** (TACAS'2007), Braga, Portugal. Part of ETAPS'2007. Topics include: rigorously based tools and algorithms for the construction and analysis of systems; formal methods, software and hardware verification, static analysis, programming languages, software engineering, real-time systems, etc. Deadline for submissions: October 6, 2006 (abstracts), October 13, 2006 (papers)

June 12th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2007), Dundee, Scotland, UK.

☺June 09-16 3rd **History of Programming Languages Conference** (HOPL-III), San Diego, CA, USA. Co-located with FCRC'2007. Deadline for submissions: August 2006 (reworked full papers)

♦June 25-29 12th **International Conference on Reliable Software Technologies - Ada-Europe'2007**, Geneva, Switzerland. Sponsored by Ada-Europe, in cooperation with ACM SIGAda (approval pending). Deadline for submissions: November 6, 2006 (papers, tutorials, workshops)

June 25-29 27th **International Conference on Distributed Computing Systems** (ICDCS'2007), Toronto, Canada. Topics include: all aspects of distributed and parallel computing. Deadline for submissions: August 15, 2006 (workshops), November 20, 2006 (papers)

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

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

2008

June 13th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2008), Madrid, Spain

SIGAda 2006

The Annual International Conference on the Ada Programming Language

Albuquerque, New Mexico, USA
12-16 November 2006

http://www.sigada.org/conf/sigada2006/

SIGAda 2006

International Conference on Software Development for Safety, Security, and High Reliability Systems

12-16 November 2006, Albuquerque, NM, USA

Sponsored by ACM SIGAda

Constructing high reliability software is an engineering challenge that can now be met in many domains. The application of software engineering methods, tools, and languages interrelate to make the challenge easier or more difficult. This conference focuses on safety, security and high reliability systems and the issues related to their development. Topics such as applied software engineering principles, conforming to specific safety or security standards, testing philosophies, programming language selection, etc. will be discussed. The conference will gather industrial experts, educators, software engineers, and researchers interested in developing, analyzing, and certifying reliable, cost-effective software. Technical or theoretical papers as well as experience reports with a focus on Ada will be presented. Contributions were received from among the following areas:

- Safety, security and high integrity development issues
- Language selection for a high reliability system: Ada, C, C++, Java, or others
- Use of high reliability subsets or dialects: Java HIP, MISRA C, Ravenscar, SPARK, etc.
- High reliability standards and their issues: DO-178B, EIC 61508, FDA, SAE, CC, EAL, etc.
- Process and quality metrics
- Analysis, Testing, and Validation
- Use of ASIS for new Ada tool development
- Mixed-language development
- Quality Assurance
- Performance analysis
- High reliability software engineering education
- High reliability development experience reports
- Real-time networking/quality of service guarantees
- Fault tolerance and recovery
- Distributed system load balancing
- Static and dynamic code analysis
- Debugging complex systems
- Integrating COTS software components
- System Architecture & Design
- Information Assurance in the age of terrorism
- Improvements and additions to the Ada language in Ada 2005
- Ada products evaluated per Common Criteria, Protection Profiles or Security Targets

The keynote address will be given by Judith Klein of Lockheed Martin. Judith will present the "Use of Ada in Lockheed Martin for Air Traffic Management and Beyond". Judith Klein is a certified systems architect at Lockheed Martin Transportation and Security Solutions. She has 28 years' experience developing distributed, real-time systems of various sizes in different domains; the last 15 years have been focused on air traffic control.

The SIGAda 2006 Program Committee is:

Program Chair	Leemon C. Baird III	leemon.baird@usafa.af.mil
Tutorials Chair	David Cook	dcook@aegistg.com
Conference Co-Chairs	Greg Gicca	gicca@ghs.com and
	Ricky Sward	ricky.sward@usafa.af.mil

See the SIGAda 2006 Home Page for further details on the conference:

http://www.acm.org/sigada/conf/sigada2006

Call for Papers

12th International Conference on Reliable Software Technologies – Ada-Europe 2007

25-29 June 2007, Geneva, Switzerland

http://www.ada-europe.org/conference2007.html

Conference Chair

Nabil Abdennadher
University of Applied Sciences, Geneva, Switzerland.
nabil.abdennadher@eig.ch

Program Co-Chairs

Nabil Abdennadher
University of Applied Sciences, Geneva, Switzerland.
nabil.abdennadher@eig.ch

Fabrice Kordon
University Pierre & Marie Curie. France
Fabrice.kordon@eig.ch

Tutorial Chair

Dominik Madon
University of Applied Sciences, Geneva, Switzerland.
dominik.madon@eig.ch

Exhibition Chair

Neville Rowden
Siemens Switzerland
neville.rowden@eig.ch

Publicity Chair

Ahlan Marriot
White-elephant, Switzerland
Alan.Marriott@eig.ch

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

Local Chair

Régis Boesch
University of Applied Sciences, Genava, Switzerland.
regis.boesch@eig.ch

In cooperation with SIGAda

(approval pending)

General Information

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

Schedule

06 November 2006	Submission of papers, workshop/tutorial proposals
26 January 2007	Notification to authors
26 February 2007	Camera-ready papers required
25-29 June 2007	Conference

Topics

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

For papers, tutorials, and workshop proposals, the topics of interest include, but are not limited to:

- **Methods and Techniques for Software Development and Maintenance**: Requirements Engineering, Object-Oriented Technologies, Formal Methods, Re-engineering and Reverse Engineering, Reuse, Software Management Issues.
- **Software Architectures**: Patterns for Software Design and Composition, Frameworks, Architecture-Centric Development, Component and Class Libraries, Component-Based Design.
- **Enabling Technology**: CASE Tools, Software Development Environments and Project Browsers, Compilers, Debuggers, Run-time Systems.
- **Software Quality:** Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems.
- **Critical Systems**: Real-Time, Distribution, Fault Tolerance, Information Technology, Safety, Security.
- **Distributed Systems**: Reliability, Security, Trust and Safety in Large Scale Distributed Platforms.
- **Mainstream and Emerging Applications**: Multimedia and Communications, Manufacturing, Robotics, Avionics, Space, Health Care, Transportation.
- **Ada Language and Technology**: Programming Techniques, Object-Oriented, Concurrent and Distributed Programming, Evaluation & Comparative Assessments, Critical Review of Language Enhancements, Novel Support Technology, HW/SW platforms.
- **Experience Reports**: Experience Reports, Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics, Experience Reports on Education and Training Activities with bearing on any of the conference topics.

Ada Conference UK 2006

K Fairlamb
AdaCore, 8 rue de Milan, Paris 75009, France; email: sales@adacore.com

Figure 1 John Rowlands, BAE Systems, presents at Ada Conference UK 2006

Abstract

March 2006 saw the welcome return of an Ada event in the UK. The Ada Conference UK 2006, operated by the Centre for Software Reliability (CSR) in collaboration with the Safety-Critical Systems Club, took place this year on 28 March at the award winning Lowry Hotel in the heart of Manchester.

Overview of the event

The focus of the event was the recent language revision, called Ada 2005, and its continued suitability for building systems where reliability, efficiency, and safety are critical. The success of the 2006 event, attracting around 120 attendees (maybe the largest professional Ada event in the world), proved that Ada is more than ever at the front of software developers' minds.

As the CSR pointed out in its announcement of the event, Ada continues to prove itself as the answer for many of today's most complex programming challenges – especially in the areas of real time, embedded and safety-critical applications and in particular as the need for robust and reliable software systems increases.

The event provided an excellent opportunity for members from all sectors of the Ada community, both in the UK and from abroad, to meet, share ideas, and reinforce links. Ada professionals from all four corners of the UK were present with a wide range of industries represented.

The event included plenary sessions by eminent Ada experts Robert Dewar and John Barnes, plus a series of technical talks by leading industrial experts, the abstracts of which are provided below and videos of which can be found on the AdaCore website at www.adacore.com.

In addition, a stream of well-attended vendor talks ran in parallel to the technical talks and a broad range of leading Ada toolset and service vendors displayed their technologies in the exhibition hall.

The next Ada UK conference is already in preparation, so look out for a forthcoming announcement regarding dates and venue!

Conference papers at Ada UK

- **Welcome to Ada 2005**

  John Barnes, author of 'Programming in Ada 2005'

  Ada 2005 is the latest chapter in the Ada story. Ada 95 was a huge leap forward from Ada 83. However, experience has shown that Ada 95 has a number of roughish edges. Ada 2005 is not such a giant leap forward but aims rather to round off Ada 95 and so provide the community with a really smooth programming language suited for the demanding applications of the 21st century. John explained the specific goals of the development and introduced the key new features of Ada 2005 and thus set the scene for the rest of the day.

- **OOP & structure control in Ada 2005**

  Pascal Leroy, IBM

  Object-oriented techniques and structure control are important in very large systems in providing flexibility and extensibility. This talk gave an overview of the numerous enhancements that have been made in this area as part of the Ada 2005 Amendment. These enhancements include topics such as: Java-like interfaces, which allow proper multiple inheritance and integrate OOP with concurrent programming; the prefixed notation, used by many other languages, which simplifies usage of complex OO architectures; type extensions in nested scopes, which make it possible to declare controlled types at any level; object factories, which make it possible to dynamically create objects of any type in a class; explicit syntax for controlling overriding, which improves the safety of OO programs; the addition of limited and private with clauses, which support mutually dependent type structures crossing package boundaries and allow finer-grain visibility control; and finally improved aggregates and function returns which make limited types more flexible and easier to use.

- **Programming & certifying Ada software on an ARINC 653 platform**

  George Romanski, Verocel Inc.

  Ada applications running in a partitioned Integrated Modular Avionics environment such as ARINC 653 constrain the programmer, but also provide greater flexibility. The Ada Tasking model may be replaced by the Process, Semaphore, Blackboard, Event and other synchronization and control mechanisms. Exception management if present, must co-exist with a Health Monitoring system. Processor-time, memory and shared resources must be robustly partitioned. This is accomplished through a configuration control mechanism. While this restricts what a programmer can do within a partition, an application may be split across several partitions, and different variants of the applications may co-exist on the same IMA platform. Multiple schedules and mode switches will then select which sets of applications should run and how transitions occur.

  An IMA system needs to be configured very carefully. Platform providers, system integrators and application developers must set up a contracting model which specifies the responsibilities for and ownership of system parameters. In a safety critical system such contract models are subject to the same certification criteria as the application programs themselves. As systems evolve and applications change, the cost of system upgrade will remain high unless the components, Ada and programs in other languages, can be treated as applications in this modular system. This reduction of cost will only be accomplished if the impact of change can be isolated to the components that change.

- **Real time issues**

  Alan Burns, University of York

  Ada 2005 has introduced a number of new features that aid the programming and analysis of real-time systems. These features include: the inclusion of the Ravenscar profile for safety critical real-time systems, CPU monitoring and accounting, budgeting for the execution time of groups of tasks, timing event for efficient time driven computation, and new scheduling polices. The latter policies being non-premeption, round robin, EDF (Earliest Deadline First) and combinations of these policies. This talk reviewed all of these features and included examples of use.

- **Building safety-critical/certified applications with Ada**

  Rod White, MBDA

  Developing safety-critical and certified applications presents different sets of problems in different domains. This talk considered those that relate to the missile products of MBDA, typically characterized by a small platform, demanding performance and a harsh environment. It considered issues such as the use of Ada, runtime systems, software re-use and the role of off-the-shelf elements. It also considered the challenges for the future – Ada has been the preferred language for a considerable period, but it is

becoming necessary to address the need to incorporate elements in other languages e.g. C – this introduces a new set of issues and concerns.

- **Demonstrating Safety-Critical properties of an automatic train protection system**

  Robin Messer, Westinghouse

  This presentation described work done in collaboration with Aerosystems International and showed how safety critical properties of an ATP have been:

  - Captured from hazard analysis
  - Analysed using a UML model
  - Translated in to SPARK annotations
  - Metrics captured on the work

- **Safety-Critical Software: Looking for an argument**

  Carl Sandom, iSys Integrity

  This presentation provided software developers with a broad overview of what an Independent Safety Auditor (ISA), safety regulator or third-party might look for when evaluating safety-critical software. The presentation should be of interest to anyone undertaking either safety-critical software development from the beginning or the retrospective safety assessment of software which has not been developed explicitly for safety-critical use but is subsequently used within safety-critical systems.

  Software safety assurance can be provided to a third party by constructing a clear and compelling safety argument which is underpinned with evidence from various diverse sources. The structure of the safety argument will determine the type and depth of the evidence that must be generated during development and/or collected in-service to support any claims made regarding the safety of the software in the context of its actual or assumed use.

  The provision of safety assurance was the central topic of this presentation and a pragmatic approach to the construction of a clear and compelling software safety argument was described in detail. The presentation was based upon a software safety assurance strategy that has been used to support system safety certification or acceptance for various real-life software development projects which the presenter has been directly involved with either as the ISA or as part of the safety assurance team.

- **Executable Modelling with UML and Ada: The X Factor**

  John Rowlands, BAE Systems

  Traditionally, executability is a property possessed by programming languages, but often not by design languages. For instance a simple UML design only captures the structure of a software system and provides a high level description of behaviour, enabling ease of navigation for maintenance. However, in order to improve the productivity of the software process, a rich model is needed that allows animation and code generation. Animation allows the design to be tested prior to committing to code or deploying to a particular platform. Full code generation allows the software to be maintained at the design level, lifting the level of abstraction at which the developer interacts with the design. However, if we are to maintain our software at the model level, we need to have access to all the features we have come to take for granted with traditional programming languages, such as ease of static checking, debugging and testing.

  In order to enrich a UML model for executability and code generation, an action language is needed. This language needs to understand the architectural concepts inherent in UML and add a detailed definition of the behaviour of the software. In the Ada community we are used to the idea that the programming language inherently provides support for finding errors early, such as strong typing, declaration before use and ease of static analysis. The ideal action language should allow the software engineer to work at the UML level of abstraction whilst providing similar static checking facilities.

  The presentation addressed the question of how such an action language could be constructed, the features that it should exhibit and the way in which it could be defined.

- **Mixed Criticality**

  Peter Amey, Praxis High Integrity Systems

  High integrity applications, such as those performing safety or security critical functions, are usually built to conform to standards such RTCA DO-178B or UK Def Stan 00-55. Typically such standards define ascending levels of criticality each of which requires a different and increasingly onerous level of verification. It is very common to find that real systems contain code of multiple criticality levels. For example, a critical control system may generate a non-critical usage log. Unless segregation can be demonstrated to a very high degree of confidence, there is usually no alternative to verifying all the software components to the standard required by the most critical element, leading to an increase in overall cost. The presentation described the novel use of static analysis to provide a robust segregation of

differing criticality levels, thus allowing appropriate verification techniques to be applied at the subprogram level. We call this fine-grained matching of verification level to subprogram criticality smart certification.

- **Ada 2005 & high integrity systems**

  Robert Dewar, AdaCore

  Ada has been, and continues to be, successful for Safety-Critical applications. This talk covered the foundations of the Ada language and its evolution being based on good programming practice and smooth integration of new features rather than specific technical capabilities. Among these readability, the package structure, the strong typing system, compile time checking, and run time exceptions all help to ensure that Ada continues to be used widely in Safety-Critical applications. The presentation concluded by emphasizing the importance of the Ada "culture" instilled in programmers.

Ada Market in 2005 Entails at Least a $5.6 Billion Investment

Ann S. Brandon

Onyons, Inc., P.O. Box 294, Randolph Center, VT 05061 USAD; e-mail: ann@onyons.com

Abstract

The Ada Resource Association sponsored a survey in 2005 on its website, adaic.org. Through the 188 responses it received in a month as well as individual interviews, it was able to gauge that the Ada market includes an investment of about $5.6 billion in Europe and North America.

Keywords: Ada market, Ada Resource Assoc., Ada.usage

1 Introduction

An international nonprofit organization, the Ada Resource Association comprises principal suppliers of Ada development environments and tools: AdaCore, IBM Rational Software, Praxis High Integrity Systems Ltd., and SofCheck. In May through early June 2005, the ARA sponsored a survey on its www.adaic.com website, and announced the results at Ada Europe 2005.

According to individual Ada corporate leaders who were interviewed and the 188 web survey responses that the ARA received, the Ada market is robust, with a total investment of at least $5.6 billion in Europe and North America. The survey asked about both current Ada usage and familiarity with or plans for Ada 2005. It was completed by software developers from North America, Australia, Korea, and almost every country in Europe.

2 322 Million LOAC, Prices and Projects Varied

The main results on Ada usage, presented at the Ada Europe conference in York, England, in June 2005, can be summarized as follows:

* Around 322 million lines of Ada code (LOAC) are in software that is either still in development or has been completed, representing a reported (and conservatively-estimated) value of around $5.6 billion. The number of LOAC is admittedly one that depends on how each company counts a line of code.

* The prices for the systems also cover a wide range. At one extreme, several software projects undertaken by volunteers or hobbyists showed zero as their cost. And at the other end of the spectrum, a response for one of the major system developments reported a cost of $2 billion.

* The projects represent a variety of applications and stages of development. (In the table below, the percentages add up to more than 100% since some respondents checked off more than one category, such as "fielded" and "maintenance"):

Project type		
Embedded systems:	44	21%
Command & Control:	32	17%
Other Types:	32	17%
Tools:	30	16%
Simulation Projects:	30	16%
Graphics:	21	11%
Libraries:	11	6%
IT Projects:	7	4%
Project Stage		
Planning:	8	4%
Development:	78	41%
Complete:	31	16%
Fielded:	54	29%
Maintenance:	56	30%
Other stage:	13	7%

2.1 Projects Outside the Defence/Aerospace Box

Although Ada's traditional stronghold has been in the defence/aerospace industry, the responses to the survey show that the language has a much broader appeal. This is likely due to Ada's intrinsic merit in helping produce reliable software and to the availability of quality Ada compilers and tools. Some of the more interesting application areas include many surprises.

2.2 List of Projects Atypical for Ada Applications

The following projects were reported by businesses and hobbyists. Some were from software engineers within businesses who prefer to program in Ada and therefore also programmed “side software” in the language they were using for their main work.

* Accounting
* Banking & Finances
* Bible Studies
* Book Title Image Matching

* Commercial Imaging
* Court Workflow
* Currency Trading
* Database Tools
* DNA Analysis
* Electronic Voting Machine
* Industrial Control
* Interlingual Machine Translator
* Internet Security
* Medical Devices & Testing
* Neuroscience Research
* Photonic Materials Research
* Security Assessment
* Semiconductor Factory
* Small Office Applications
* Spellcheck
* Tension Structure Analysis
* Warehouse Management/Control

3 Understanding of the Ada 2005 Standard Features Better than Expected

The survey also collected data on respondents' acquaintance with and usage plans for features that are being added to the Ada 2005 modification.

The survey offered six possible answers for each feature that it presented, from "Unaware" and "Do not understand" to "Frequently use". The following specific features were listed: "limited with"; interfaces; scheduling improvements; the container library; nested extensions; prefixed views; directories/environment/calendar packages; enhanced anonymous access types; limited aggregates and functions; overriding indicators; Ravenscar; and expanded Unicode support. A roughly one-line description was given of each.

Since tutorial or rationale material on the language modification had only recently been made available to the general Ada community, and since most of the information available had been highly technical, a deep understanding of the new features would have been somewhat surprising. The actual results — on average, about 34% of the respondents either didn't answer the question, or said that they either were unaware of a feature or didn't understand it — are probably better than expected and reflect a high degree of interest in the new amendment to the Ada language.

3.1 New Ada 2005 Features Likely to be Used

The best understood new features were the containers library and the other new packages, while the least understood feature was overriding indicators. Unlike most of the other features, the description in the survey's question on overriding indicators didn't explain their use, which might have explained respondents' confusion.

Those features that respondents said they would never use proved to be highly specialized. Further, if a feature were understood, it would tend to be used: on average, more than 80% of the users who understood a feature said that they would use it at least occasionally.

An interesting counterexample was the Ravenscar Profile: 32% of the respondents that understood the feature said that they would never use it. This may seem surprising, since the Ravenscar profile is generally regarded as one of Ada's major strengths for high-integrity applications. But most of the survey's respondents are working on systems that, although requiring high reliability, are not safety critical. The developers can thus use the full Ada language rather than a specialized subset.

The feature most likely to be used by developers who indicated an understanding of the feature is the new standard packages (for directories/environment variables/calendar), followed by the containers library, prefixed views, and overriding indicators.

Conclusion

The Ada market is robust, especially in the embedded systems and command and control software, for which the language was designed.

As for the survey's results concerning users' knowledge of and interest in the Ada 2005 modification, they seem to validate the ARG effort in choosing how to update the language. The survey's data show a higher degree of familiarity with the new features than expected, and reveal even before the Ada community had access to a formal Rationale that the new libraries were already considered the language's most useful addition.

Ada-Europe Awards

Year	Best Paper	Best Presentation	Proceedings
2006	**Benjamin M. Brosgol** and **Andy Wellings** "*A Comparison of Ada and Real-Time Java™ for Safety-Critical Applications*"	**Sri Narayanan** "*Secure Execution of Computations in Untrusted Hosts*" **Michael Ward** "*Parallel Graphical Processing in Ada*"	Pinho, Luís Miguel, González Harbour, Michael (Eds.): Reliable Software Technologies – Ada-Europe 2006: Proceedings 11th Ada-Europe International Conference on Reliable Software Technologies, Porto, Portugal, June 5-9, 2006, LNCS(4006) Springer-Verlag 2006. ISBN: 3-540-34663-5
2005	**Peter Amey**, **Rob Chapman**, and **Neil White** "*Smart Certification of Mixed-Criticality Systems*"	**Michael Gonzáles-Harbour** "*RT-EP: A Fixed-Priority Real-Time Communication Protocol over Standard Ethernet*"	Vardanega, Tullio, Wellings, Andy (Eds.): Reliable Software Technologies – Ada-Europe 2005: Proceedings 10th Ada-Europe International Conference on Reliable Software Technologies, York, United Kingdom, June 20-24, 2005. LNCS(3555) Springer-Verlag, 2005. ISBN 3-540-26286-5
2004	**Alan Burns**, **Andy J. Wellings** and **S. Tucker Taft** "*Supporting Deadlines and EDF Scheduling in Ada*" **Peter Amey** and **Neil White** "*High Integrity Ada in a UML and C World*"	**Adrian J. Hilton** "*High-Integrity Interfacing to Programmable Logic with Ada*"	Albert Llamosi, Alfred Strohmeier (Eds.): Reliable Software Technologies - Ada-Europe 2004: Proceedings 9th Ada-Europe International Conference on Reliable Software Technologies, Palma de Mallorca, Spain, June 14-18, 2004. LNCS(3063) Springer-Verlag, 2004. ISBN 3-540-22011-9
2003	**Miguel Masmano**, **Jorge Real**, **Ismail Ripoll** and **Alfons Crespo** "*Running Ada on Real-Time Unix*"	**Jorge Real** "*Running Ada on Real-Time Unix*"	Jean-Pierre Rosen, Alfred Strohmeier (Eds.): Reliable Software Technologies - Ada-Europe 2003: Proceedings 8th Ada-Europe International Conference on Reliable Software Technologies, Toulouse, France, June 16-20, 2003. LNCS(2655), Springer-Verlag, 2003. ISBN 3-540-40376-0
2002	**Robert Dewar**, **Olivier Hainque**, **Dirk Craeynest** and **Philippe Waroquiers** "*Exposing Uninitialized Variables: Strengthening and Extending Run-Time Checks in Ada*"	**Jean-Pierre Rosen** "*Ada, Interfaces and the Listener Paradigm*"	Johann Blieberger, Alfred Strohmeier (Eds.): Reliable Software Technologies - Ada-Europe 2002: Proceedings 7th Ada-Europe International Conference on Reliable Software Technologies, Vienna, Austria, June 2002. LNCS(2361) Springer-Verlag, 2002. ISBN 3-540-43784-3

Year	Best Paper	Best Presentation	Proceedings
2001	**Alexandre Duret-Lutz** "*Expression Templates in Ada*"	**Alan Burns** "*Defining New Non-Preemptive Dispatching and Locking Policies for Ada*"	Dirk Craeynest, Alfred Strohmeier (Eds.): Reliable Software Technologies - Ada-Europe 2001: Proceedings 6th Ada-Europe International Conference on Reliable Software Technologies, Leuven, Belgium, May 2001. LNCS(2043) Springer-Verlag, 2001. ISBN 3-540-42123-8
2000	**Andy J. Wellings**, **Robert W. Johnson**, **Bo I. Sanden**, **Jörg Kienzle**, **Thomas Wolf** and **Stephen Michell** "*Object-Oriented Programming and Protected Objects in Ada 95*"	**Brian Dobbing** "*Using JavaTM APIs with Native Ada Compilers*" (paper co-authored with Shayne Flint)	Hubert B. Keller and Erhard Plödereder (Eds.): Reliable Software Technologies - Ada-Europe 2000: Proceedings 5th Ada-Europe International Conference on Reliable Software Technologies, Potsdam, Germany, June 26-30, 2000. LNCS(1845) Springer-Verlag, 2000. ISBN 3-540-67669-4
1999	**Jorge Real** and **Andy Wellings** "*The Ceiling Protocol in Multi-Moded Real-Time Systems*"	**Roderick Chapman** "*Re-engineering a safety-critical application using SPARK 95 and GNORT*" (paper co-authored with Robert Dewar)	Michael González Harbour, Juan A. de la Puente (Eds.): Reliable Software Technologies - Ada-Europe'99: Proceedings LNCS(1622) Springer-Verlag, 1999. Ada-Europe International Conference on Reliable Software Technologies, Santander, Spain, June 7-11, 1999. ISBN 3-540-66093-3
1998	**Agustín Espinosa**, **Vicente Julián**, **C. Carrascosa**, **Andrés Terrasa** and **Ana García-Fornes** "*Programming Hard Real-Time Systems with Optional Components in Ada*"	**Brian Dobbing** "*The Ravenscar Tasking Profile for High Integrity Real-Time Programs*" (paper co-authored with A. Burns and G. Romanski)	L. Asplund (Ed.): Reliable Software Technologies - Ada-Europe'98: Proceedings LNCS(1411) Springer-Verlag, 1998. Ada-Europe International Conference on Reliable Software Technologies, Uppsala, Sweden, June 8-12, 1998. ISBN 3-540-64536-5
1997	**Michael Gonzalez-Harbour**, **Javier J. Gutiérrez García** and **J.C. Palencia Gutiérrez** "*Implementing Application-Level Sporadic Server Schedulers in Ada 95*"	**Alex E. Bell** "*An Alternative Toolset for Analysis of Ada Programs*"	Keith Hardy, Jim Briggs (Eds.): Reliable Software Technologies - Ada-Europe'97: Proceedings. LNCS(1251) Springer-Verlag, 1997. Ada-Europe International Conference on Reliable Software Technologies, London, UK, June 2-6, 1997. ISBN 3-540-63114-3
1996	**David E. Emery**, **Richard F. Hilliard II**, **Timothy B. Rice** "*Experiences Applying a Practical Architectural Method*"	**Wolfgang Gellerich** "*Where Does the GOTO Go To?*" (paper co-authored with Markus Kosiol and Erhard Plödereder)	Alfred Strohmeier (Ed.): Reliable Software Technologies - Ada-Europe'96: Proceedings. LNCS(1088) Springer-Verlag, 1996. Ada-Europe International Conference on Reliable Software Technologies, Montreux, Switzerland, June 10-14, 1996. ISBN 3-540-61317-X

Ada-Europe 2006 Sponsors

AdaCore
Contact: Zépur Blot
8 Rue de Milan, F-75009 Paris, France
Tel: +33-1-49-70-67-16 Fax: +33-1-49-70-05-52
Email: sales@adacore.com URL: www.adacore.com

Aonix
Contact: Jacques Brygier
66/68, Avenue Pierre Brossolette, 92247 Malakoff, France
Tel: +33-1-41-48-10-10 Fax: +33-1-41-48-10-20
Email : info@aonix.fr URL : www.aonix.com

Green Hills Software Ltd
Contact: Christopher Smith
Dolphin House, St Peter Street, Winchester, Hampshire, SO23 8BW, UK
Tel: +44-1962-829820 Fax: +44-1962-890300
Email : URL : www.ghs.com

I-Logix
Contact: Martin Stacey
1 Cornbrash Park, Bumpers Way, Chippenham, Wiltshire, SN14 6RA, UK
Tel: +44-1249-467-600 Fax: +44-1249-467-610
Email : info_euro@ilogix.com URL : www.ilogix.com

Praxis High Integrity Systems Ltd
Contact: Rod Chapman
20 Manvers Street, Bath, BA1 1PX, UK
Tel: +44-1225-466-991 Fax: +44-1225-469-006
Email : sparkinfo@praxis-his.com URL : www.sparkada.com

Ellidiss Software
TNI Europe Limited
Contact: Pam Flood
Triad House, Mountbatten Court, Worrall Street, Congleton, CW12 1DT, UK
Tel: +44-1260-29-14-49 Fax: +44-1260-29-14-49
Email: info@tni-europe.com URL: www.ellidiss.com