ADA USER JOURNAL

Volume 30
Number 1
March 2009

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

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

Reprinted Articles

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

Editorial

This first issue of Volume 30 marks the beginning of the celebration of the Thirty Years of the Ada User Journal. The longevity of the Journal is a credit to the Journal authors and readers, after all the reason for the Journal's existence, that have demonstrated during these years their loyalty and interest. Also important, it is a credit to all that volunteered their effort in the preparation, publishing and in the distribution of the Journal. To all of them, I address our appreciation.

I am very much pleased that the first article of this issue, kindly provided to us by John Barnes, is an overview of the history of the Ada User Journal during these thirty years. As a special treat, the article presents images of some of the covers of the Journal during its different periods, which are exceptionally published in color in order that our readers can fully appreciate them. I know that the readers will enjoy reading the article as much as I did.

As a side note, I would like to point our readers to the online archive of the Journal, available via the website of Ada-Europe, where the latest period in the history of the Ada User Journal can be browsed online. At this time, the online archive provides the full contents of all issues from March 2002 to March 2008, and the table of contents of the last four published issues.

Coming back to this issue, it also provides a paper by James Moore and John Benito, with a progress report on the ISO/IEC Technical Report 24772 concerning Programming Language Vulnerabilities, a work being done in the framework of the Vulnerabilities Working Group (WG 23). I take this opportunity to point out that this matter will also be analyzed, obviously with a focus in Ada, in a one day workshop planned to be held in conjunction with the Ada-Europe 2009 conference, this June in Brest, France.

The third paper of the issue finalizes the publication of material derived from the Industrial Track of the Ada-Europe 2008 conference. The paper, by Bradley Moore, of General Dynamics, Canada, explores some of the Ada 2005 features and the Distributed Systems Annex to support complex and dynamic networks.

The technical part of the issue ends with two interesting Ada Gems, by Matthew Heaney and Quentin Ochem, related to Interfaces. And as usual, the reader will find the rich information of the News and Calendar sections, contributed by Marco Panunzio and Dirk Craeynest, their respective editors.

Luís Miguel Pinho
Porto
March 2009
Email: lmp@isep.ipp.pt

News

Marco Panunzio

University of Padua. Email: panunzio@math.unipd.it

Contents

Ada-related Events

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

FOSDEM 2009 — Presentations available on-line

From: Dirk Craeynest <dirk@aqua.cs.kuleuven.be>
Date: Thu, 12 Feb 2009 14:07:21 +0100 CET
Subject: FOSDEM 2009 - Presentations Ada Developer Room on-line
Newsgroups: comp.lang.ada,fr.comp.lang.ada, comp.lang.misc,be.comp.os.linux

---

All presentations available on-line

Ada Developer Room at FOSDEM 2009

(Free and Open-Source Software Developers' European Meeting)

Saturday 7 & Sunday 8 February 2009

Université Libre de Bruxelles (U.L.B.), Brussels, Belgium

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/09/090207-fosdem.html

Organized in cooperation with Ada-Europe

All presentations at the Ada Developer Room, held at FOSDEM 2009 in Brussels last weekend, are available on the Ada-Belgium web site now.

A few pictures taken during the event are on-line as well.

Most presentations can be downloaded in an open format (ODP) and for all a PDF version is available:

- "Introduction to Ada", Jean-Pierre Rosen, Adalog
- "GNAT Project Facility", Vincent Celier, AdaCore
- "GPS - GNAT Programming Studio", Vincent Celier, AdaCore
- "Ada in Debian GNU/Linux", Ludovic Brenta, Debian
- "Ada Distributed Systems Annex", Thomas Quinot, AdaCore
- "NARVAL - Distributed Data Acquisition in Ada", Xavier Grave, Centre National de la Recherche Scientifique
- "GPRBuild - Multi-Language Builder", Vincent Celier, AdaCore
- "OO Programming Model in Ada 2005", Jean-Pierre Rosen, Adalog
- "Ast2Cfg - Framework for CFG-Based Analysis and Visualisation of Ada Programs", Georg Kienesberger, Vienna University of Technology
- "MaRTE-OS - Minimal Real-Time Operating System for Embedded Applications", Miguel Telleria de Esteban, Universidad de Cantabria
- "MaRTE-OS - Practical View", Daniel Sangorrin, Universidad de Cantabria
- "GNATBench - Ada Eclipse plugin", Vincent Celier, AdaCore

For these presentations and pictures, see:

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/09/090207-fosdem.html

Thanks again to all presenters for their collaboration and thanks to the many participants from all over Europe for their interest!

Ada-Belgium on Internet

Home: http://www.cs.kuleuven.be/~dirk/ada-belgium

FTP: ftp://ftp.cs.kuleuven.be/pub/Ada-Belgium

E-mail: ada-belgium-board@cs.kuleuven.be

Mailing list: ada-belgium-info-request@cs.kuleuven.be

AdaCore — Ada-Europe 2009

From: AdaCore Press Center
Date: Tuesday February 17, 2009
Subject: Ada Europe 2009
RSS: http://www.adacore.com/2009/02/17/ada-europe-2009/

AdaCore is a major sponsor of this event. AdaCore staff will also be giving the following talks:

- Implementation of the Ada 2005 Task Dispatching Model in MaRTE OS and GNAT - Jose F. Ruiz, Mario Aldea, Michael Gonzalez Habour.

The following tutorials:

- Building Cross Language Applications using Ada - Quentin Ochem
- Hard Real-Time and Embedded Systems Programming - Pat Rogers
- Software Fault Tolerance - Pat Rogers

And the following industrial presentation:

- Couverture - An Innovative Open Framework for Coverage Analysis of Safety Critical Applications - Matteo Bordin

AdaCore — Embedded World

From: AdaCore Press Center
Date: Friday February 20, 2009
Subject: Embedded World
RSS: http://www.adacore.com/2009/02/20/embedded-world/

AdaCore will be exhibiting at this event.

AdaCore — Facing The New Challenges Of Your Embedded Architecture

From: AdaCore Press Center
Date: Friday February 20, 2009
Subject: Facing The New Challenges Of Your Embedded Architecture
RSS: http://www.adacore.com/2009/02/20/facing-the-new-challenges-of-your-embedded-architecture/

AdaCore will be exhibiting at this event and Michael Friess will be a member of the Avionics Panel Discussion.

AdaCore — Lean Event

From: AdaCore Developer Center
Date: Thursday February 5, 2009

Subject: The lean, agile approach to high-integrity programming
RSS: http://www.adacore.com/2009/02/05/lean-agile-event/

AdaCore is organizing a one day conference in Paris, March 26, around the theme "The Lean, Agile Approach to High Integrity Software". Our panel of experts will present real world examples that illustrate how 'Lean Production' and 'Agile Programming' concepts are being successfully applied to software development.

Speakers include:

- Jim Sutton (Author of Lean Software Strategies)
- Roberto di Cosmo - University of Paris VII
- Alexandre Boutin - Yahoo
- Emmanuel Chenu - Thales
- David Jackson / Andy Vickers - Praxis High-Integrity Systems
- Cyrille Comar - AdaCore

For more information on this event, please visit:

http://www.adacore.com/lean-event

If you, and/or any of your colleagues would like to register free of charge for the event, please send an email to events@adacore.com.

[see also "AdaCore — Lean Event" in AUJ 29-4 (Dec 2008), p.228. —mp]

Praxis HIS — ATC Global 2009

From: Praxis Press Center
Date: Tuesday February 24, 2009
Subject: Praxis at ATC Global - 17th to 19th March 2009
URL: http://www.praxis-his.com/news/index.asp

Praxis will be exhibiting at ATC Global on 17-19 March 2009 at the Amsterdam Rail Exhibition and Congress Centre.

Praxis HIS — Ada UK 2009

From: Praxis Press Center
Date: Tuesday February 24, 2009
Subject: Praxis at Ada UK - 24th March 2009
URL: http://www.praxis-his.com/news/index.asp

Praxis is attending and exhibiting at the Ada Conference UK to be held on 24th March 2009 in London. Speakers from Praxis will include Rod Chapman, Product Manager for SPARK and Janet Barnes one of Praxis' key technical authorities.

Ada-related Resources

DIANA

From: Patrick Boulay <boulayp@free.fr>
Date: Wed, 17 Dec 2008 21:46:16 -0800
Subject: DIANA
Newsgroups: comp.lang.ada

I am looking for a reference on the DIANA so that I can understand its definition, workings, layout, use etc. I have searched on the net extensively and not found a downloadable definition or description of it.

Does anyone know a public link where I can download documents on DIANA?

From: Markus Schoepflin <nospam@no.spam>
Date: Thu, 18 Dec 2008 10:37:29 +0100
Subject: Re: DIANA
Newsgroups: comp.lang.ada

Not a downloadable reference, but maybe you can get your hands on this book in a library near you:

Ada Software Tools Interfaces: Workshop, Bath, July 13-15, 1983.

Proceedings (Lecture Notes in Computer Science) (v. 180) (Paperback)

See http://www.amazon.com/Ada-Software-Tools-Interfaces-Proceedings/dp/3540138781/ref=sr_11_1?ie=UTF8&qid=1229592865&sr=11-1

[…]

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 18 Dec 2008 10:23:30 +0100
Subject: Re: DIANA
Newsgroups: comp.lang.ada

Diana was an attempt to standardize the intermediate tree of Ada compilers, in order to ease the development of 3rd party tools. Some compilers (Rational IIRC) still use Diana, but the goal is now achieved by the ASIS interface.

What kind of application do you have in mind?

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 18 Dec 2008 13:29:00 +0100
Subject: Re: DIANA
Newsgroups: comp.lang.ada

[…]

> PL/SQL reverse engineering

Although there is a clear filiation from Ada to PL/SQL, they are different languages, and it is not possible to use Ada tools on PL/SQL.

I doubt also that PL/SQL uses Diana trees...

Of course, if you want to learn Diana just to see what a syntactic tree looks like, that's fine.

#Ada IRC Channel on Freenode

From: Caracal <caracal@niestu.com>
Date: 07 Feb 2009 05:45:57 GMT
Subject: #Ada IRC channel on Freenode
Newsgroups: comp.lang.ada

This is the annual reminder of the existence of the #Ada channel on the Freenode IRC network. Now over seven years old, the channel is open to all discussions related to the Ada language and its use. We welcome beginners and pros alike, and do our best to maintain a friendly, productive, and informative atmosphere. The channel has been growing steadily; we now average about 40 members and have active conversations daily.

Point your IRC client to irc.freenode.net and join the #Ada channel.

[…]

Ada and Wikipedia

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 24 Feb 2009 12:03:47 +0100
Subject: Invade wikipedia!
Newsgroups: comp.lang.ada

I was musing recently through Wikipedia. There are lots of topics related to algorithms or similar stuff that have examples, generally in C, often with examples in some other languages for comparison.

It would be nice if the community joined forces to add Ada examples. Nobody can do that alone, but if everyone add some examples when finding an opportunity, we could give much more visibility to Ada.

Public Ada Library

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Thu, 26 Feb 2009 18:45:54 +0100
Subject: Re: HELP TO BEGINNER
Newsgroups: comp.lang.ada

[…]

> Exploring the WEB I've found the PAL (Public Ada Library) but it's unavailable. I'm interested to know if there is some free archive containing various code (algorithms about numerical integration and similar...) […]

http://www.iste.uni-stuttgart.de/ps/AdaBasis/pal_1195/ada/pal.html

[…]

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Fri, 27 Feb 2009 01:03:13 -0800 PST
Subject: Re: HELP TO BEGINNER (PAL)
Newsgroups: comp.lang.ada

[…] Here is the link to the *partial* mirror:

ftp://ftp.cs.kuleuven.ac.be/pub/Ada-Belgium/mirrors/pal/

I gonna make a backup right now, before this one disappears the same way as all the others…

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 27 Feb 2009 18:11:08 -0600
Subject: Re: HELP TO BEGINNER (PAL)
Newsgroups: comp.lang.ada

Please don't worry about that; there is a copy in the AdaIC archives:

http://archive.adaic.com/ase/index.html

This is a copy of Dirk's FTP site, except of course it is available through HTTP, not FTP, so you can browse it and find it from Google.

There isn't much chance of AdaIC disappearing as long as Ada exists, so I wouldn't worry about it disappearing. (We have it in the archives as it has not been maintained for years, so there is some stale stuff in it, like ancient versions of GNAT).

From: Dirk Craeynest <dirk@heli.cs.kuleuven.be>
Date: Mon, 2 Mar 2009 18:30:02 +0000 UTC
Subject: Re: HELP TO BEGINNER (PAL)
Newsgroups: comp.lang.ada

[…] Note that the latest version of the *full* Public Ada Library is available at

ftp://ftp.cs.kuleuven.ac.be/pub/Ada-Belgium/cdrom/index.html.

[…]

The copy at the Ada-Belgium server dates from the period when distributing "big files" via http was a no-no, and people insisted that ftp should be used for that instead. Since then the situation changed "somewhat", so making a copy available via http is no longer a problem and in fact may be preferable for many reasons […].

Nevertheless, I received reports from people who appreciate the availability via ftp, e.g. because they have only ftp- and no http- access, so I intend to keep it online as long as cs.kuleuven.be offers an ftp-service.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 2 Mar 2009 16:17:11 -0600
Subject: Re: HELP TO BEGINNER (PAL)
Newsgroups: comp.lang.ada

[…]

It's good to have both. I agree that it is better to get large files by FTP. The main problem with having the whole thing hosted on FTP is that it makes it hard to browse the HTML index pages. So the best possible solution would be to have it split between HTTP and FTP -- small files and indexes in HTTP and large files in FTP -- but as that is a lot of work, I didn't want to bother with it.

Probably the best approach is to find the files that you need in the AdaIC archives and then download them from Dirk's FTP site if they are large.

(Filezilla works well for accessing FTP sites.)

Reference Implementation of Design Patterns in Ada

From: ceretullis
Date: Tuesday, Mar 3 2009 16:31
Subject: Design Pattern reference implementations in Ada?
URL: http://stackoverflow.com/questions/607126/design-pattern-reference-implementations-in-ada

Does anyone know of some good reference implementations of common design patterns in Ada?

From: Marc C
Date: Tuesday, Mar 3 2009 17:13
Subject: Design Pattern reference implementations in Ada?
URL: http://stackoverflow.com/questions/607126/design-pattern-reference-implementations-in-ada

Several articles about "Software Patterns Implemented in Ada".

[see http://www.adapower.com/index.php?Command=Class&ClassID=Patterns&Title=Patterns —mp]

Ada-related Tools

Simple Components 3.3

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 13 Dec 2008 16:58:25 +0100
Subject: ANN: Simple components version 3.3
Newsgroups: comp.lang.ada

The current version provides implementations of smart pointers, sets, maps, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, re-entrant mutexes, deadlock-free arrays of mutexes), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support. It grew out of needs and does not pretend to be universal. Tables management and strings editing are described in separate documents see Tables and Strings edit.

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

Changes to the previous version:

- The procedure Purge and the function Is_Preserved were added to the packages Generic_FIFO and Generic_FIFO.Generic_Signaled in order to remove undesired FIFO elements;
- References of blackboards are made comparable;
- Bug fix in blackboards. The bug manifested itself as ghost elements appearing under certain circumstances.

AVR-Ada 1.0

From: Rolf Ebert <rolf_ebert@users.sourceforge.net>
Date: Sun, 7 Dec 2008 00:21:10 -0800 PST
Subject: [Announce] AVR-Ada V1.0 released
Newsgroups: comp.lang.ada, comp.arch.embedded

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

The project description consists of some wiki pages at:

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

This is a source only release, i.e. you get patches and instructions for building your own cross compiler, the run time system, and the support packages.

There are no pre-built binaries yet.

The download is available at the green button on page

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

If you have difficulties in building or using the compiler or you want to chat about a project, please join and use the mailing list at

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

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 run-time system
- a useful AVR specific support library
- support packages for accessing LCDs and for Dallas' 1-wire sensors

The current distribution of AVR-Ada is v1.0. It is based on gcc-4.3.3.

The Ada run time system (RTS) on the other hand is still not even a *run* time system. It is more a compile time system. Most 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 a bit of AVR specific support. Some type and interface definitions,

timing routines, eeprom access, UART, and most importantly the necessary definitions for most AVR parts. The syntax for accessing ports and specific bits changed radically (compared to previous releases) thanks to a patch provided by AdaCore.

You can now read and write every predefined port either as an Unsigned_8 or as an array (0..7) of Boolean.

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

GWenerator 0.95

From: Gautier de Montmollin <gdemont@users.sourceforge.net>
Date: Mon, 08 Dec 2008 21:43:21 +0100
Subject: Ann: GWenerator 0.95
Newsgroups: comp.lang.ada

The first graphical version of GWenerator is out!

Visit: http://sf.net/projects/gnavi/

GWenerator allows doing Graphical User Interfaces with existing software like Visual Studio or the free ResEdit (http://resedit.net) and program Windows applications in Ada using the GWindows object-oriented library.

GWenerator produces Ada sources corresponding to dialogs and menus, as a background task. Of course, GWenerator's own GUI is itself produced this way - kind of a self-demo.

The archive contains some other examples and numerous stress-tests downloaded from Internet.

But the better is to play around and send feedbacks or ask questions on the GNAVI mailing list…

PCSC/Ada 0.5

From: Reto Buerki <reet@codelabs.ch>
Date: Sat, 13 Dec 2008 12:48:46 +0100
Subject: [Announce] PCSC/Ada version 0.5 released
Newsgroups: comp.lang.ada

I'm proud to announce the first public release of PCSC/Ada.

The PCSC/Ada library provides bindings to PC/SC-middleware for the Ada programming language. PCSC/Ada allows you to communicate with smart cards using the SCard API with Ada.

Besides the thin and thick bindings to PC/SC, the actual version contains the API documentation for PCSC/Ada version 0.5, code examples, unit- and integration test suites.

For more information on PCSC/Ada, visit:

http://www.nongnu.org/pcscada/

Test reports, code reviews and of course patches are much appreciated!

PLplot Ada bindings

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Fri, 2 Jan 2009 15:31:10 -0800 PST
Subject: ANN: Ada bindings to PLplot plotting software
Newsgroups: comp.lang.ada

This is to announce the official availability of the Ada language bindings to the plotting package PLplot as of version 5.9.1. See the home page at

http://plplot.sourceforge.net/

Available unofficially for some time, the Ada bindings are now enabled by default.

PLplot is an actively developed project at SourceForge and should be considered for high quality publication-grade plotting. It is a linkable library and thus potentially faster than some other packages, not requiring writing data to a file before plotting.

From the PLplot home page:

"PLplot is a cross-platform software package for creating scientific plots. To help accomplish that task it is organized as a core C library, language bindings for that library, and device drivers which control how the plots are presented in non-interactive and interactive plotting contexts."

"The PLplot core library can be used to create standard x-y plots, semi-log plots, log-log plots, contour plots, 3D surface plots, mesh plots, bar charts and pie charts. Multiple graphs (of the same or different sizes) may be placed on a single page, and multiple pages are allowed for those device formats that support them."

The Ada bindings offer a number of substantial improvements over the C API, including a choice of binding, one using "traditional" PLplot names such as plcol0(1) to set the pen color to red, the other using "Ada-friendly" names such as Set_Pen_Color(Red).

The Ada bindings are extended to offer a host of "simple plotters" that do not require user set-up and will suffice for many day-to-day plotting needs.

Here is an example:

```
with
  PLplot_Auxiliary,
  PLplot;
use
  PLplot_Auxiliary,
  PLplot;

procedure Simple_Example is
  x, y : Real_Vector(-10 .. 10);
begin
  for i in x'range loop
    x(i) := Long_Float(i);
    y(i) := x(i)**2;
  end loop;

  Initialize_PLplot; -- Call this only once.
  Simple_Plot(x, y); -- Make the plot.
  End_PLplot;        -- Call this only once.
end Simple_Example;
```

There is also extensive Ada-specific documentation.

Platforms Supported:

- Linux, Mac OS X, and other Unix-based
- Windows (2000, XP and Vista)
- MS-DOS (DJGPP)

Output File Formats:

- CGM
- GIF
- JPEG
- LaTeX
- PBM
- PDF
- PNG
- PostScript
- SVG
- Xfig

Interactive Platforms:

- X
- GNOME
- Tcl/Tk
- PyQt
- wxWidgets

Language Bindings:

- Ada
- C/C++/D
- Fortran 77/90
- Java
- OCaml
- Octave
- Perl
- Python
- Tcl/Tk
- wxWidgets

From: Georg Bauhaus <see.reply.to@maps.futureapps.de>
Date: Sat, 03 Jan 2009 01:34:01 +0100
Subject: Re: ANN: Ada bindings to PLplot plotting software
Newsgroups: comp.lang.ada

Skimming the PLplot examples, which are given for several languages (http://plplot.sourceforge.net/examples.php), I noticed that all examples mirror the fortranesque C library

functions more or less directly. (6 char names with a "pl" prefix making them 8 char, tons of numeric parameters; I understand that "these examples also serve as a testbed for the bindings in Ada and other languages by checking the Postscript files that are generated by each example against those generated by the C versions.")

However, authors of some bindings including this Ada one have added "a number of substantial improvements over the C API". (Set_Pen_Color(Red) seems so much better than plcol0(1), even when it does not name the thing that is turned red; I don't want to sound complaining, in particular having done nothing on this work). That's somewhat hidden from the web presentation, though, I think.

[…]

I'm still looking for an alternative for some Perl-driven GNUplot command scripts, so I'm really happy to see PLplot growing into the post-70s era.

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Mon, 5 Jan 2009 15:15:51 -0800 PST
Subject: Re: ANN: Ada bindings to PLplot plotting software
Newsgroups: comp.lang.ada

Read the docs. I originally called this procedure Set_Pen_Color(..) but changed it when I figured that not everyone appreciates the reference to pen plotters and would respond, "What pen?" Calling this procedure simply causes everything that is drawn thereafter to be in the specified color, until the procedure is called again with a different argument.

> Can I put an item on the Plplot binding wish list? That would be: show some examples that use the thick bindings, on the web pages. Maybe these examples using thick bindings will then help some demonstrate why they are using a language that offers more than Fortran 77 style C.

I couldn't agree with you more. In fact, I developed the "Ada-name" binding first and made the "traditional-name" later at the request of the PLplot developer team.

I'm very happy to report that I took your petition to the developers and you should soon see Ada examples written with both bindings appearing on the PLplot web site with the "Ada-name" versions given preference if possible. I'll post back here when that upgrade happens.

From: Jerry Bauck <lanceboyle@qwest.net>
Subject: Re: ANN: Ada bindings to PLplot plotting software
Date: Mon, 5 Jan 2009 23:56:00 -0800 PST
Newsgroups: comp.lang.ada

Done. Check it out at

http://plplot.sourceforge.net/examples.php?demo=3D01.

QtAda 2.1.0

From: Vadim Godunko <vgodunko@gmail.com>
Date: Wed, 14 Jan 2009 02:33:04 -0800 PST
Subject: Announce: QtAda 2.1.0
Newsgroups: comp.lang.ada

We are pleased to announce the immediate availability of the QtAda 2.1.0 version. You can download multi platform source code package or Microsoft Windows binary package from our download page:

http://www.qtada.com/en/download.html

QtAda is an Ada 2005 language bindings to the Qt libraries and a set of useful tools. QtAda allows easily to create cross-platform powerful graphical user interface completely on Ada 2005. QtAda applications will work on most popular platforms - Microsoft Windows, Mac OS X, Linux/Unix - without any changes and platform specific code. QtAda allows to use all power of visual GUI development with Qt Designer.

New in QtAda 2.1.0:

- documentation in the Qt Assistant format was added;
- integration with the GNAT tool chain including gprbuild and GPS was added;
- support for the QtXml modules was added;
- support for the QtCore and QtGui modules was extended;
- support for the safe pointers to the instances of the QObject subclasses was added;
- support for printing was added.

Commercial support and services are available from the QtAda team.

QtAda/GtkAda integration kit

From: Vadim Godunko <vgodunko@gmail.com>
Date: Sat, 21 Feb 2009 08:57:11 -0800 PST
Subject: Announce: QtAda/GtkAda integration kit
Newsgroups: comp.lang.ada

We are pleased to announce a technology preview for the new product - QtAda/GtkAda Integration Kit. It allows to create hybrid QtAda/GtkAda application, thus it allows to migrate and reuse of the existent GtkAda code in QtAda application.

Technology preview is available for immediate download from the QtAda download page (see snapshot section):

http://www.qtada.com/en/download.html

QtAda/GtkAda Integration Kit can be used with latest QtAda 2.2.0 snapshots only.

TASH 8.4

From: Pascal Obry <pascal@obry.net>
Date: Sat, 17 Jan 2009 13:17:04 +0100
Subject: TASH 8.4
Newsgroups: comp.lang.ada

[…] I'm looking for the Tash 8.4 binding. The www.adatcl.com site seems down, do you know where to download this package? Found nothing with Google... […]

From: Graham Stark <graham.stark@virtual-worlds.biz>
Date: Sat, 17 Jan 2009 06:11:45 -0800 PST
Subject: Re: TASH 8.4
Newsgroups: comp.lang.ada

[…]

http://tcladashell.wiki.sourceforge.net/

From: okellogg <okellogg@freenet.de>
Date: Tue, 20 Jan 2009 11:45:41 -0800 PST
Subject: Re: TASH 8.4
Newsgroups: comp.lang.ada

[…] Note that the last release, 20081009, still had a number of problems related to the Ada interfacing representations for Tcl C data structures.

These have been corrected in the current svn trunk (r121).

TASH 20090207

From: sjw <simon.j.wright@mac.com>
Date: Sat, 7 Feb 2009 12:34:32 -0800 PST
Subject: TASH 20090207
Newsgroups: comp.lang.ada

We're pleased to announce the release of TASH 20090207, at https://sourceforge.net/project/showfiles.php?group_id=164395.

This release includes a source release and a Windows binary release (Windows 2000, GNAT-GPL-2008, ActiveState Active Tcl 8.4.17.0).

The readme:

This is TASH version 8.6-0, intended for use with Tcl/Tk versions 8.4 through 8.6.

The 'thick binding' (packages Tash.*) depends on several symbols that are hidden in Tcl later than 8.4 if compiled with GCC later than 4.0. For this reason, the thick binding is, by default, not built if setup.tcl finds that Tcl is later than 8.4.

In order to use the thick binding with Tcl later than 8.4 and GCC later than 4.0, you need to build Tcl and Tk from source and edit the following entry from the unix/Makefiles:

```
AC_FLAGS   = ... lots of flags ... and:
   -DMODULE_SCOPE=extern\
__attribute__\(\(__visibility__\(\"hidden\"\
)\)\)
```

The part

-DMODULE_SCOPE=extern\
__attribute__\(\(__visibility__\(\"hidden\"\
)\)\)

should be removed from the AC_FLAGS of both Tcl and Tk.

Then do "make clean && make" and then "make install".

You can then edit the value of SUPPORTS_TASH in the setup.tcl dialog and build TASH.

We would like to remove the dependency on Tcl private symbols in a future version of TASH. However, we suspect that the actual use of these facilities is limited "in the field".

API changes since previous release:

- Updated the Tcl Ada API to native Tcl 8.[4-6]
- Removed function Tcl.Tcl_GetObjId as it is not part of native Tcl and it gave trouble on 64-bit systems.
- Many of the record types in package Tcl were declared private although their native Tcl counterpart is public. Certain functions could not even be used without public access to the record type's contents. These types are now public.
- Removed type Tcl.Tcl_Obj_Ptr as it is not part of native Tcl, and its use was prone to introduce memory leaks.

Ada for Netbeans

From: Andrea Lucarelli <andrea.lucarelli@gmail.com>
Date: Sun, 28 Dec 2008 08:13:43 -0800 PST
Subject: Ada for Netbeans
Newsgroups: comp.lang.ada

[…]

The first version (very alpha release) of Ada for Netbeans is accessible at this link

http://plugins.netbeans.org/PluginPortal/faces/PluginDetailPage.jsp?pluginid=13977.

A lot of features are not available or only draft, see

http://wiki.netbeans.org/Ada

for more details on these features.

From: Andrea Lucarelli <andrea.lucarelli@gmail.com>
Date: Tue, 30 Dec 2008 05:52:04 -0800 PST
Subject: Re: Ada for Netbeans
Newsgroups: comp.lang.ada

[…] Unfortunately I tested Ada for Netbeans only on Windows systems. […]

From: Xavier Grave <xavier.grave@ipno.in2p3.fr>
Date: Tue, 30 Dec 2008 10:57:56 +0100
Subject: Re: Ada for Netbeans
Newsgroups: comp.lang.ada

[…] With Netbeans 6.5 all the installation process is smooth, great. […]

From: Andrea Lucarelli <andrea.lucarelli@gmail.com>
Date: Mon, 5 Jan 2009 08:00:21 -0800 PST
Subject: Re: Ada for Netbeans
Newsgroups: comp.lang.ada

I solved the following problems in the 0.1.6 version

(http://plugins.netbeans.org/PluginPortal/faces/PluginDetailPage.jsp?pluginid=3D13977):

1. Fix string and char literal recognition.

2. Fix new Ada platform and auto-detect in Linux system.

3. Fix gpr template file for Linux systems.

4. Fix project actions for Linux systems.

[…]

From: Xavier Grave <xavier.grave@ipno.in2p3.fr>
Date: Mon, 05 Jan 2009 20:50:47 +0100
Subject: Re: Ada for Netbeans
Newsgroups: comp.lang.ada

[…] I have tested your last version on a Debian GNU/Linux with custom directory for GNAT (GPL 2007). It worked near perfection. The only problem (very little) I encountered was when I tried directly a clean and build project.

[…]

From: Andrea Lucarelli <andrea.lucarelli@gmail.com>
Date: Sun, 1 Feb 2009 16:32:33 -0800 PST
Subject: Re: Ada for Netbeans [0.1.6]
Newsgroups: comp.lang.ada

[…] I just published a new version of Ada for Netbeans (0.1.7):

http://plugins.netbeans.org/PluginPortal/faces/PluginDetailPage.jsp?pluginid=3D13977

It's an incremental version. I have also corrected a pair of bugs from your comments […] and to improve the auto-detect, I also check the environment path variable now.

[…]

From: Andrea Lucarelli <andrea.lucarelli@gmail.com>
Date: Wed, 4 Feb 2009 13:21:37 -0800 PST
Subject: Re: Ada for Netbeans [0.1.7]
Newsgroups: comp.lang.ada

[…] Access-to-subprogram is an Ada 2005 extension.

My parser for now is compatible only with the syntax of Ada 95. […]

From: Andrea Lucarelli <andrea.lucarelli@gmail.com>
Subject: Re: Ada for Netbeans [0.1.7]
Date: Sun, 8 Feb 2009 01:38:06 -0800 PST
Newsgroups: comp.lang.ada

[…]

I activated on SourceForge the possibility to communicate bugs, and support the request for new features:

http://sourceforge.net/tracker/?group_id==3D248853.

To collaborate as developer it is necessary to follow the information that you find in this page:

http://www.netbeans.org/community/contribute/project-and-contrib.html

[…]

MacPorts / GNU Ada Release

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Tue, 17 Feb 2009 18:50:36 +0100
Subject: New MacPorts / GNU Ada Release
Newsgroups: comp.lang.ada

I have released a new MacPort version of the GCC compiler [1]. The Portfiles for the new release has been send up stream as I have become a MacPorts maintainer [2] and they can be compiled and installed like any other Port (MacPorts is a source code based distribution).

Since you need a bootstrap compiler I also created a binary release [3] to get everybody started.

And on top of that I prepared new source code tarballs for booch95 and XML/Ada.

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

[2] http://trac.macports.org/wiki/MacPortsDevelopers

[3] https://sourceforge.net/project/showfiles.php?group_id=12974&package_id=291480

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Thu, 19 Feb 2009 11:19:33 +0100
Subject: Re: New MacPorts / GNU Ada Release
Newsgroups: comp.lang.ada

[…]

All I did is to copy the port of "gcc43" and added Ada to it. In particular, I made sure that a

gvimdiff gnat-gcc/Portfile gcc43/Portfile

gives me a manageable result so work on GCC upgrades are minimized.

So any question related to non-Ada stuff should be directed to the author of gcc34 which is mww (Markus W. Weissmann, http://trac.macports.org/wiki/mww).

Sorry to spoil your fun but until there is an Objective-Ada the various Objective runtimes are not high on my priority list.

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Tue, 17 Feb 2009 12:32:16 -0800 PST

Subject: Re: New MacPorts / GNU Ada Release
Newsgroups: comp.lang.ada

[…]

Can I now have a working 4.3.x GNAT on a PPC Mac running OS X 10.5.x? As you know, this combination has not been provided by MacAda.org. I have been running their GNAT on my PPC Powerbook under OS X 10.4 but have been unable to switch to 10.5 because of this "missing compiler".

If I understand correctly, due to your bootstrapping efforts, I can use my existing MacAda compiler as the bootstrap by specifying +macada (and presumably +ada and +powerpc).

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Wed, 18 Feb 2009 09:20:39 +0100
Subject: Re: New MacPorts / GNU Ada Release
Newsgroups: comp.lang.ada

Have a look at my bootstrap job:

http://gnuada.svn.sourceforge.net/viewvc/gnuada/trunk/MacPorts/Utilities/Build_GCC.command?view=markup

Well an old version - there is a commit still pending - I commit the correct version tonight. Anyway, you can get the draft here. Just replace \<.*gcc43\> with gnat-gcc.

[see also “MacPorts” in AUJ 29-4 (Dec 2008), p.234. —mp]

Zip-Ada

From: Gautier de Montmollin <gdemont@users.sourceforge.net>
Date: Sun, 11 Jan 2009 19:48:42 +0100
Subject: Ann: UnZip-Ada v.27
Newsgroups: comp.lang.ada

Big improvement as from v.26: the zip archive can be any kind of stream, not only a file - this for compressing and decompressing. It allows to use the library for the most various purposes (doing everything in memory, chaining streams, etc.). Visit:

http://unzip-ada.sf.net/

From: Gautier de Montmollin <gdemont@users.sourceforge.net>
Date: Sat, 07 Feb 2009 20:39:19 +0100
Subject: Ann: UnZip-Ada v.30
Newsgroups: comp.lang.ada

Some news from the [Un]Zip-Ada library @ http://unzip-ada.sf.net/ :

Main changes from versions '28' to '30':

- Added support for the 64KB-slide "Enhanced deflate" format #9 in UnZip.Decompress
- Added Find_Zip tool (search through an archive)
- Added Demo_csv_into_zip demo (production of numerous files into a zip archive, only one temp file)
- Added Zip.LZ77 and Zip.Compress.Reduce
- Added an ./extra directory with a tiny LZH encoder/decoder
- ZipAda uses Zip.Create

From: Gautier de Montmollin <gdemont@users.sourceforge.net>
Date: Mon, 23 Feb 2009 20:57:54 +0100
Subject: Ann: Zip-Ada v.31
Newsgroups: comp.lang.ada

Some news from the Zip-Ada library @ http://unzip-ada.sf.net/ :

- the library's name is now Zip-Ada and not UnZip-Ada anymore (except the site name at SF which is in the process of being renamed…)
- Added tiny demos: Demo_Zip, Demo_UnZip
- ! Zip.Create: Create / Finish: if Info.Stream is to a file, the underlying archive file is also created / closed as well
- Added procedure Add_String in Zip.Create

(- ! marks an incompatibility)

[…]

[See also “UnZip-Ada” in AUJ 29-1 (Mar 2009), p.10. —mp]

Ahven — Unit Test Library

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Thu, 22 Jan 2009 22:17:48 +0200
Subject: ANN: Ahven 1.4
Newsgroups: comp.lang.ada

I am pleased to announce Ahven 1.4.

Ahven is a simple unit test library for Ada 95 distributed under permissive ISC license.

This release introduces Test Anything Protocol (TAP) reporter, a new API for stack-based test cases, and improved Janus/Ada support. Also, some API changes were done, but they should affect only those who have extended the framework.

For more info, please read the detailed release notes:

http://home.gna.org/ahven/release_1_4.txt

or visit Ahven's homepage:

http://home.gna.org/ahven/

The source code is available as tar.gz and zip packages:

http://download.gna.org/ahven/ahven-1.4.tar.gz

http://download.gna.org/ahven/ahven-1.4.zip

Known issues:

- Some test cases fail with Janus/Ada 3.1.1d, but the TAP reporter and the framework should still work as intended. (Later compiler versions are ok.)
- Ahven as a dynamic library might not work with GNAT GPL 2008. (Static library or GNAT GPL 2007 should be ok.)

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Mon, 23 Feb 2009 22:44:31 +0200
Subject: ANN: Ahven 1.5
Newsgroups: comp.lang.ada

I am pleased to announce Ahven 1.5.

Ahven is a simple unit testing library for Ada 95 programming language.

Ahven project recently moved from Gna! to SourceForge and this is the first release from sf.net.

Changes include:

- Rewritten Janus/Ada build system
- Minor changes to GNAT build scripts
- Bug fix for Ahven.Slist package API documentation generation

New homepage location is

http://ahven.sourceforge.net/

Source code is available as tar.gz and zip packages from

http://sourceforge.net/project/showfiles.php?group_id=253736

Version control repository can be found from

http://ahven.cvs.sourceforge.net/ahven/

From: Rolf Ebert <rolf_ebert@users.sourceforge.net>
Date: Mon, 23 Feb 2009 14:50:48 -0800 (PST)
Subject: Re: ANN: Ahven 1.5
Newsgroups: comp.lang.ada

[…]

What's the difference to aunit?

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue, 24 Feb 2009 07:47:21 +0200
Subject: Re: ANN: Ahven 1.5
Newsgroups: comp.lang.ada

Ahven's API is similar to AUnit (1.0x series), but Ahven is plain Ada 95 code and works with other compilers than GNAT. In addition, Ahven is distributed under ISC (modified BSD) license, so you can embed it freely into commercial applications.

You can also get test results in XML format and therefore integrate Ahven into continuous integration systems (Cruisecontrol, Hudson).

Originally, I created Ahven because AUnit was missing some features which I needed and instead of extending pure GPL project, it was easy enough to create a new library with ISC license.

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Tue, 24 Feb 2009 11:38:56 +0100
Subject: Re: ANN: Ahven 1.5
Newsgroups: comp.lang.ada

[…]

> In addition, Ahven is distributed under ISC (modified BSD) license, so you can embed it freely into commercial applications.

More accurately, please consider replacing "commercial" with "protected source" or similar.

GPL software is the technical fundament of a number of commercially successful enterprises.

I do know that not all software can be exposed freely to the many freeloading business men out there. Which is what the GPL will require of those selling binary programming produce.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 24 Feb 2009 09:45:53 -0800 PST
Subject: Re: ANN: Ahven 1.5
Newsgroups: comp.lang.ada

> More accurately, please consider replacing "commercial" with "protected source" or similar.

I'd rather call it "secret source" or simply "proprietary". To me "protected source" means that I am protected because I can see, modify and distribute the source, i.e. it is Free Software

> GPL software is the technical fundament of a number of commercially successful enterprises.
>
> I do know that not all software can be exposed freely to the many freeloading business men out there. Which is what the GPL will require of those selling binary programming produce.

Yes and the GNU Affero General Public License (AGPL) version 3 extends the requirement to distribute the sources to those using the "software as a service" business model.

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue, 24 Feb 2009 23:02:22 +0200
Subject: Re: ANN: Ahven 1.5
Newsgroups: comp.lang.ada

Thanks for the suggestions. I actually use term "commercial application" only in the manual (which is not built or installed by default), but I changed the wording there to "proprietary commercial application" [1], just in case someone looks at it.

[1] Wikipedia seems to use this form in its BSD license article:

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

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Sat, 28 Feb 2009 11:53:11 +0200
Subject: ANN: Ahven 1.6
Newsgroups: comp.lang.ada

[…]

I messed up GNAT installation in version 1.5.

Thanks to Reto Buerki for noticing it and providing a fix.

I released today Ahven 1.6, which is basically Ahven 1.5 with the GNAT installation fix applied.

Like earlier, source code is available as tar.gz and zip packages from

http://sourceforge.net/project/showfiles.php?group_id=253736

[See also "Ahven — Unit test library" in AUJ 29-3 (Sep 2008), p.156. —mp]

Ada-related Products

AdaCore — GNAT Pro 6.2.1

From: AdaCore Press Center
Date: Tuesday February 17, 2009
Subject: GNAT Pro 6.2.1
RSS: http://www.adacore.com/2009/02/17/gnat-pro-621/

GNAT Pro 6.2.1 is a major release comprising 18 native and 28 cross platforms qualified for 146 different environments (including variations of Ada runtimes as well as host and target OS versions). For a full list of supported platforms, please visit:

www.adacore.com/home/gnatpro/supported_platforms

The core compiler back-end technology is now based on a gcc 4.3, while the core debugging engine is based on gdb 6.8. These upgrades allow significant performance gains as well as porting to new architectures.

New supported platforms in 6.2.1 include:

- x86_64-darwin
- jvm-windows
- x86-elinos-linux
- avr-elf-windows
- leon-elf-windows

An internal AdaCore ACATS certificate is now included for relevant platforms. It is a first step in providing more reviewable qualification evidence as part of the GNAT Pro release. Note that ACATS testing represents only one component of our overall qualification procedures.

Main native platforms now offer 2 additional components that were beta-tested last year:

- Ada-Java Interfacing Suite (AJIS)
- GNAT Component Collection (GNATcoll)

This new release incorporates fixes for the all issues that have been reported to AdaCore as documented in the 'Known Problems' files of the GNAT Tracker

Documentation section: Known problems in GNAT Pro 6.1.2.

In the same section, you can also find 'Features' entries describing all the new features and enhancements. Major ones include:

- Improved support for safety-critical applications
  - o Traceability to source code
  - o Coverage analysis Ability to associate pre- and post-conditions with subprograms
- Ability to selectively enable or disable groups of assertions
- Additional rules in gnatcheck for coding standard verification
- More efficient implementation of stack checks, overflow checks, and validity checks
- Additional attributes and pragmas to ease generic programming
- Communication-related improvements
  - o More efficient string streaming
  - o Better support for serial communication and socket handling Additional compilation warnings on suspected errors
- Missing overriding indicators
- Assumption that string lower bound is 1.

The 6.2.1 release also includes an enhanced version of the GNAT Programming Studio (GPS) IDE. GPS 4.3.x is compatible with GNAT Pro versions 3.16a1 up to 6.2

[For AJIS, see http://www.adacore.com/home/gnatpro/toolsuite/ada_java/

For GNATcoll, see http://www.adacore.com/2008/06/17/gnat_component_collection/ —mp]

Aonix — ObjectAda for VxWorks

From: Aonix Press Release
Date: Wednesday February 4, 2009
Subject: Aonix Releases ObjectAda for Wind River VxWorks RTOS
URL: http://www.aonix.com/pr_02.04.09.html

ObjectAda Full Standard Ada Runtime Executes atop VxWorks 6.6

San Diego, February 4, 2009—Aonix®, the provider of a provider of solutions for safety and mission-critical applications, today announced the release of ObjectAda® 8.4 for Windows, targeting PowerPC embedded and real-time systems running the Wind River VxWorks 6.6 real-time operating system

(RTOS). This is the first ObjectAda release supporting full Ada tasking atop VxWorks 6.6 via Real-Time Processes (RTP).

ObjectAda for VxWorks leverages Wind River Workbench, an Eclipse-based development environment providing developers access to the broad range of tools available through the Eclipse framework. With support for multiple operating systems, architectures and programming languages, ObjectAda for VxWorks provides the flexibility to standardize on a single development framework. Users also have the option to utilize ObjectAda's standard graphical or command-line interface. The ObjectAda compilation system is comprised of an integrated language-sensitive editor, source-code browser, compiler with industry-leading compilation speed, debugger and full library manager.

Embedded systems development often begins without target hardware in hand. In the absence of a PowerPC execution platform, VxSim, a target simulation facility supplied in the VxWorks distribution, can perform initial application execution and testing direct from the Intel/Windows host development platform. This is especially cost-efficient when multiple developers vie for access to expensive and scarce target hardware testing cycles.

"Ada development is a strong aerospace and defense requirement," said Rob Hoffman, vice president and general manager for Aerospace and Defense at Wind River. "Continued support and development of systems can often span decades and we're pleased to have up-to-date and well-integrated solutions for our customers."

"Demand for Ada products has remained remarkably strong, even in the face of current economic uncertainty," commented Gary Cato, Aonix director of marketing. "Our traditional customers are looking for ways to modernize or upgrade their applications in cost-effective ways. The new release of ObjectAda Real-Time for VxWorks 6.6 provides a great vehicle to achieve these objectives. "

About the ObjectAda Family

ObjectAda is an extensive family of native and cross development tools and runtime environments. ObjectAda native products provide host development and execution support for the most popular environments including Windows, Linux and various Unix operating systems. ObjectAda Real-Time products provide cross development tools on Windows, Linux or Unix systems which target PowerPC and Intel target processors running in a full Ada "bare" runtime or in conjunction with popular RTOSs. ObjectAda RAVEN® products provide a hard real-time Ada runtime to address those systems requiring certification to the highest levels of safety standards such as DO-178B Level A for flight safety.

Shipping and Availability

ObjectAda Real-Time targeting Power Architectures running Wind Rivers' VxWorks 6.6 is immediately available starting at $15,000 in the U.S. with quantity discounts available.

About Aonix®

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

For more information, visit

www.aonix.com.

RainCode — RainCode Checker 2.0

From: RainCode Press Release
Date: November 2008
Subject: Releases of RainCode Checker 2.0
URL: http://www.raincode.com/index.html

Version 2.0 of RainCode's coding guidelines enforcement tool The RainCode Checker will be released in January 2009.

The Checker is available for Ada, C and COBOL, and comes as an Eclipse plugin for its user interface.

Technical information

The RainCode Checker for Ada provides you with a convenient infrastructure to check automatically for compliance with company-specific or standard coding conventions in your Ada code. The Checker analyzes the Ada source files, detects where the coding rules have been violated, and generates a detailed report listing the encountered offences.

Based on the RainCode Engine's static analysis capabilities, the RainCode Checker enables you to verify simple as well a complex coding rules:

- Lexical rules: "Identifier homonymy is forbidden"
- Syntactical rules: "Nested package declaration is not allowed"
- Semantic rules: "Overloading type names in the package Standard is prohibited"
- Global rules: "Two distinct formal parameters cannot be associated with the same actual parameter"

Flexibility

- RainCode Checker for Ada verifies about 70 coding rules by default. It allows you to select the rules you actually want the RainCode Checker to check, or to use them as examples to code your own rules in the tool.
- The tool is multi-platform (Windows, Unix, Solaris, and all Unix-like)
- RainCode Checker is adaptable: each company can have its specific RainCode Checker for Ada, with its own coding guidelines.

Testing and documentation

The Checker allows you to attach, for each rule, a set of positive and negative examples that show the expected errors. The regression testing facility checks that each rule is correctly implemented, and documents what each rule does in the generated report.

Report Generation

After you have checked the whole project against a set of rules, you can ask RainCode Checker to generate a report with different levels of detail.

This report in PDF format can be used:

- as a deliverable for a third party, which lists all the sources which have been checked, with the matching list of offences; or
- as a complete documentation of the coding guidelines used within the organization or project.

Evaluation Version

The RainCode Checker for Ada comes in a user-friendly GUI version. To see what it looks like, and what it can do for you, just register here and log in. On the download page, you will have access to the evaluation version, which is built on a sample set of 20 Ada sources.

[see http://www.raincode.com/adachecker.html —mp]

Rapita Systems — Trial Version of RapiTime

From: Rapita Press Center
Date: Monday February 9, 2009
Subject: Rapita launches trial version of execution time analysis tools
URL: http://www.rapitasystems.com/trial_version_launched

Rapita Systems announced today the availability of trial version of the RapiTime execution time measurement and analysis software. The trial version is a ready-to-go kit that includes a compiler, a microcontroller simulator, example applications and the RapiTime software.

RapiTime helps developers of embedded, real-time systems to prove that their applications meet performance requirements, and pinpoints where to optimise applications that don't meet these

requirements. RapiTime makes a series of highly detailed measurements of the execution time of your application on its target hardware. Using a model derived from static analysis of the software, RapiTime analyses the timing behaviour of your embedded, real-time application to tell you:

- Performance measurements including Worst-Case Execution Time (WCET), minimum, maximum and average execution time.
- Probability distribution of execution times of your code at different levels of detail.
- Code coverage: show which lines of code have been executed by your tests.
- Which lines of code lie on the critical path of the WCET.

RapiTime's Eclipse plug-in allows you to explore the timing analysis results at levels of detail that range from the entire application to functions or even lines of code.

"We're really excited about the trial version" said Andrew Coombes, Marketing Manager at Rapita Systems. "It provides an all-in-one solution for anyone with an interest in real-time systems to experience and explore the latest version of RapiTime".

The trial version includes a 30 day trial licence for the RapiTime v2.1 software, a tutorial guide, example applications, an ARM9 Simulator and a compiler for the simulator.

To request a copy of the evaluation version, please visit

http://www.RapitaSystems.com/evaluation

Vector Software — VectorCAST 5.0

From: Vector Software Press Release
Date: Friday February 27, 2008
Subject: Vector Software announces the formal release of VectorCAST 5.0.
URL: http://www.vectorcast.com/news/index.php

East Greenwich, RI – February 27, 2009 - Vector Software, Inc., a world leader in the embedded software test tool market, today announced the formal release of VectorCAST 5.0.

Version 5.0 highlights

Stub-by-function – The stubbing of functions can now be dynamically controlled on a per-test case basis. This means that you can create test cases for a function, and in one test case have a dependent function stubbed, while in another test case, the real code will be used. This feature does not require any rebuilding or recompilation of the test environment, and is available even for functions defined in the file or the C++ class that is under test. This allows testing to be performed on one individual function in complete isolation from its dependent functions.

Global Data Filtering – The test case editor has been enhanced to allow relationships between functions and global variables to be displayed. When the global data filtering is enabled only the objects referenced by that function will be displayed in the editor. Additionally, clicking on a function will highlight all of the global variables that are used by that function, similarly, clicking on a global variable, will highlight all of the functions that reference that variable.

CSV Data Editor – A new comma-separated-value (CSV) data editor has been added to allow test case construction from CSV files created by modeling tools or Excel spreadsheets. The editor provides the ability to open CSV files in a spreadsheet view, and map columns onto data items in the test environment by simply dragging and dropping cells onto the test case editor tree. The resultant map file can then be used to automatically create a test case for each row of data in the CSV file. Customers who are interested in upgrading to the new release should login to the customer portal of our web site, or contact their sales person at sales@vectorcast.com. All existing customers with a current maintenance contract will be sent keys for version 5.0 automatically.

About Vector Software

Vector Software, Inc is a leading independent provider of automated software testing tools. Vector Software's VectorCAST™ line of products reduces the burden placed on individual developers by automating and standardizing application-component testing. The VectorCAST™ tools support the C, C++, Ada 83, and Ada 95 programming languages. The market focus of Vector Software is on companies developing embedded systems for aerospace, military, medical, telecom, and process-control applications.

[see http://www.vectorcast.com/pdf/press-release-for-vc-50.pdf —mp]

Ada and GNU/Linux

Linux Device Drivers in Ada

From: Brian Drummond <brian_drummond@btconnect.com>
Date: Tue, 13 Jan 2009 12:38:59 +0000
Subject: Linux device drivers in Ada?
Newsgroups: comp.lang.ada

Possibly a long shot, but does anyone know of any example Linux device drivers written in Ada? Is it even possible for Linux?

I'm not looking for anything specific, rather for a starting point. I need to write a device driver for an FPGA card on the PCIe bus, and I'd rather not use C if at all possible.

From: Jacob Sparre Andersen <jspa@nykredit.dk>
Date: Tue, 13 Jan 2009 05:21:25 -0800 PST
Subject: Re: Linux device drivers in Ada?
Newsgroups: comp.lang.ada

It is definitely possible. The primitive way to do it is to compile it using GNAT and Pragma (No_Runtime). I think I've read a description of a more elegant solution. You might also want to take a look at RTL-GNAT.

(see http://rtportal.upv.es/apps/rtl-gnat/).

From: Dirk Heinrichs <dirk.heinrichs@online.de>
Date: Tue, 13 Jan 2009 18:43:51 +0100
Subject: Re: Linux device drivers in Ada?
Newsgroups: comp.lang.ada

There is an example in "The Big Online Book of Linux Ada Programming" [1].

Lookup chapter 16.16 "Writing Linux Modules".

[1] http://www.pegasoft.ca/resources/boblap/book.html

From: Brian Drummond <brian_drummond@btconnect.com>
Date: Wed, 14 Jan 2009 01:36:01 +0000
Subject: Re: Linux device drivers in Ada?
Newsgroups: comp.lang.ada

[…] RTLGnat may not help with the driver per se, but certainly will with handling modules.

Ada and Microsoft

Calling an Ada library from Visual C++

From: markp <markwork66@yahoo.com>
Date: Mon, 23 Feb 2009 08:14:44 -0800 PST
Subject: Calling Ada Library from C++
Newsgroups: comp.lang.ada

[…] I am trying to make a static Ada library that is callable from Visual C++. I have created a my_library.a file by using a GNAT project file similar to the following:

```
project My_Lib is
   for Source_Dirs use ("src1", "src2");
   for Object_Dir use "obj";
   for Library_Name use "my_library";
   for Library_Dir use "lib";
   for Library_Kind use "static";
end My_lib;
```

I am not familiar with adding libraries to Visual C++. It appears the convention is a ".lib" file vice a ".a". Am I on the right track with this? Can I simply rename my

".a" file, or use it as is? How do I link it into the C++ application? Any help provided on this would be very much appreciated.

From: Manuel Gomez <mgrojo@gmail.com>
Date: Mon, 23 Feb 2009 12:41:19 -0800 PST
Subject: Re: Calling Ada Library from C++
Newsgroups: comp.lang.ada

I cannot help you myself, but take a loot at this wikibook article, it might be useful:

http://en.wikibooks.org/wiki/Ada_Programming/Platform/Windows/Visual_C++_-_GNAT_interface

[…]

References to Publications

Why aren't developers interested in Ada?

From: Jack Ganssle <jack@ganssle.com>
Date: Wednesday February 4, 2009 [last modified on. —mp]
Subject: Why aren't developers interested in Ada?
URL: http://www.embedded.com/212902632

[A short article that tries to start a discussion about the reduced market share of Ada in the embedded domain, compared to languages like C; despite programs written in Ada exhibit fewer bugs and are delivered faster. Some selected, self-contained comments to it follow. —mp]

From: Mike Perkins
Date: Thursday January 29, 2009
Subject: Why aren't developers interested in Ada?
URL: http://www.embedded.com/212902632

One important attribute for a programming language to be successful is that it is taught in schools, with early adoption by youngsters creating a waterfall effect.

The fact is that many high schools have classes in C while Ada is rarely seen in colleges. I think that's a big factor. For wide adoption, a language needs the support of academia. I realize at this point in time, it's somewhat of a chicken-or-egg thing.

[…]

From: John McCormick
Date: Friday January 30, 2009
Subject: Why aren't developers interested in Ada?
URL: http://www.embedded.com/212902632

We still teach Ada to our students at the University of Northern Iowa. From 2000 to 2007 we taught Java in CS1 and CS2 and Ada in CS3. The faculty unanimously agreed that our use of Java for beginners was a failure. Starting in the Fall of 2007 we began two parallel tracks for Freshman. One track taught C/C++ in CS1 and CS2, the other Ada. Students now get Java in CS3. The first groups took CS3 last Fall. The CS3 teacher reported that in the final grade distribution ALL of the students with the Ada background did better than those with the C/C++ background. Numbers are too small yet to publish an education article, but it is certainly looking strong for Ada as a good language for teaching beginners.

From: BlackAmber
Date: Wednesday February 4, 2009
Subject: Why aren't developers interested in Ada?
URL: http://www.embedded.com/212902632

I'm one of those dinosaurs that was first exposed to Ada in the early 80's and fell in love with it.

[…] When it all came down to the end of the argument it seemed decisions were made on emotion rather than logic - "We can't have the government dictating how we develop programs!".

[…] For those that say that Ada isn't a good choice because there aren't enough trained developers I would counter that someone can be trained to be a proficient Ada programmer far quicker than it takes to train that same person to produce acceptable code in C++. And you can rest assured that the code produced by that Ada developer is of better quality and higher reliability that the C++.

[Read also the follow-up article by Jack Ganssle: "Ada Take Two" at http://www.embedded.com/213400901 — mp]

AdaCore — Legacy Is Not a Four-Letter Word

From: AdaCore Press Center
Date: Wednesday January 21, 2009
Subject: 'Legacy' is not a four-letter word
RSS: http://www.adacore.com/2009/01/21/legacy/

Military Embedded Systems

[see the article by Robert Dewar at http://www.mil-embedded.com/articles/id/?3729 —mp]

AdaCore — A Principled Approach to Software Engineering

From: AdaCore Developer Center
Date: Thursday February 5, 2009
Subject: A principled approach to software engineering
RSS: http://www.adacore.com/2009/02/05/a-principled-approach-to-software-engineering/

This paper examines the use of Java as a first programming language, in the light of well-established principles of software engineering, and the increasing concern with correctness, performance, and maintainability. We argue that Java is markedly inferior to Ada or C++ as a language for introductory Computer Science courses, and that its widespread use in the training of tomorrow's software engineers is counterproductive.

[see http://www.adacore.com/wp-content/uploads/2009/02/principled_approach.pdf —mp]

AdaCore — OpenCert 2009

From: AdaCore Press Center
Date: Wednesday February 18, 2009
Subject: OpenCert 2009
RSS: http://www.adacore.com/2009/02/18/opencert-2009/

Jose F. Ruiz will present the paper "Open-DO: Open Framework for Critical Systems".

Ada Inside

Indirect Information on Ada Usage

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

Job offer [Belgium]: Ada 95 Developer

My client in Brussels requires an Ada 95 developer to work on software integration and automated test processes.

Essential skills include:

- Expert knowledge of Ada 95 software development and architecture.
- Knowledge of software integration and automation test process.
- Expertise in design and development of mission critical systems in a distributed environment.
- Incident and problem investigation.
- Operating systems: Unix and Linux.
- Air traffic flow management experience.

Job offer [United Kingdom]: Ada 95, Software Engineer

[…] Defence Organisation based in Bristol is looking to recruit 3 Ada Software Engineers

The Contract is for 6 Months initially and then ongoing after at £36 Per Hour

Please Note - Candidates must be Currently Security Cleared to SC Level. Ada 95, Software Engineer – Development/test Software Engineer capable of design,

Development and test of systems for Defence applications. Knowledge of Defence Standards would therefore be useful.

The Development languages will be mainly Ada 95 (AdaCore). The software is expected to require properties associated with dependability (reliability, security), so skills associated with AdaTest, software proving etc.. It is expected that the requirements and design specification will use a mixture of the Doors requirement management system, Matlab (Simulink), Rhapsody UML and Madge/Mascot.

The configuration management system is the Dimensions tool.

Job offer [United Kingdom]: Ada Software Engineer

Our client is an Engineering company specialising in safety-critical systems based in Buckinghamshire. They're looking for a Software Engineer to join them on what will be a challenging and diverse project.

You must have proven experience in the following:

Excellent knowledge of software full life cycle including requirements analysis, architecture/design, coding and unit test as well as integration and validation.

You will require the following skills:

- Ada 95
- C / C++
- DOORS
- Rhapsody
- Object-Oriented Design
- Formal Design Methodologies UML / HOOD

Job offer [United Kingdom]: Software Engineer

[…] you will take responsibility for the development of Safety Critical software within the software engineering department, through the full life cycle. […]

Software Engineer Key Responsibilities:

- Prepare requirements specifications;
- Prepare design documentation;
- Write software code (C, C++, Ada, C#);
- Develop Test Specifications and perform tests;
- Support of Integration and System Testing.

Software Engineer Qualifications and Experience:

- Degree in computer, software engineering, electronics or mathematical subjects;
- Software engineering experience from safety-critical field (aerospace, nuclear, rail, defence, medical devices etc..);
- proficient in coding in C, C++ and Ada;
- OO Analysis;
- knowledge of networking protocols such as TCP/IP.

Job offer [United States]:

[…]

Qualifications:

- Must be able to obtain a Secret Clearance
- Bachelor of Science in Electrical Engineering or Bachelor of Science in Computer Science
- Minimum 5+ years experience in Software Development with real-time embedded experience using object-oriented design (OOD) and analysis (OOA) methodologies.
- Design and development using C, C++ or Ada
- Must be able to work in a team environment

Preferred Qualifications:

- Experience in Radar Software, VxWorks, and Mercury

Job offer [Italy]:

Graduates in Engineering or Computer Science with 1/5 years of experience for firmware development in embedded applications (Automotive, Transport and Avionics).

Requested:

- Assembly, C, C++, Ada
- RTOS (RTAI, VxWorks, Integrity, pSos, OSE..)
- Windows CE/Mobile
- DSP, PPC, micro controllers

Optional:

- SW Life Cycle
- Standards DO178-B, Cenelec 50128 (SIL), ECSS
- USB, SPI, ETH device Drivers
- Technical documentation

Ada in Context

On the Prologue and Epilogue of Ada Procedures

From: Pini <zepini@gmail.com>
Date: Tue, 16 Dec 2008 14:52:50 -0800 PST
Subject: Strip procedure prologue and epilogue
Newsgroups: comp.lang.ada

A usual procedure like

```
procedure Foo
begin
  -- some stuff here;
end;
```

generates a prologue like

```
pushl %ebx
-- generated code for some stuff here
popl %ebx
```

I have a very specific procedure whose purpose is to be called at bootloading time and set up a stack, so there is no need for the prologue and epilogue of this procedure to exist. Is there an Ada-only way (through a pragma for instance) to strip the prologue and epilogue of this procedure ?

[…]

From: Samuel Tardieu <sam@rfc1149.net>
Date: Fri, 19 Dec 2008 01:01:47 +0100
Subject: Re: Strip procedure prologue and epilogue
Newsgroups: comp.lang.ada

[…]

Probably not, as the existence of the prologue/epilogue is not supposed to change anything for the Ada programmer.

If you're using GNAT, you may be lucky by using "-fomit-frame-pointer" on your code.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 16 Dec 2008 15:06:24 -0800 PST
Subject: Re: Strip procedure prologue and epilogue
Newsgroups: comp.lang.ada

[…]

I'm not an expert in this matter but maybe you should look at how Lovelace boots. The source is in the public monotone server I just mentioned and there's a snapshot at

http://people.debian.org/~lbrenta/org.os-lovelace-2008-07-02T21:01:02.tar.gz

More info about Lovelace on

http://www.lovelace.fr

From: Rolf Ebert <rolf_ebert@users.sourceforge.net>
Date: Wed, 17 Dec 2008 00:11:58 -0800 PST
Subject: Re: Strip procedure prologue and epilogue
Newsgroups: comp.lang.ada

[…]

There is no general Ada solution for that, but a GNAT specific pragma. GCC provides the attribute "naked" for some back-ends. It is used like this:

```
procedure Foo;
pragma Machine_Attribute (
   Entity        => Foo,
   Attribute_Name => "naked");
```

The pragma is intended for start-up code like initialising some RAM area or for writing your own task switching code.

Be very careful here and inspect the generated assembler code. Naked procedures will not only omit the prologue and epilogue but also the ret

statement. You typically must not use any code that requires support from the run time system at that stage.

From: Pini <zepini@gmail.com>
Date: Wed, 17 Dec 2008 01:40:44 -0800 PST
Subject: Re: Strip procedure prologue and epilogue
Newsgroups: comp.lang.ada

[…] I was looking for something like that, but unfortunately GCC doesn't support the naked attribute on x86 (and probably never will, as it is considered harmful by GCC developers).

[…]

On Duration

From: David Henry <tfc.duke@gmail.com>
Date: Thu, 19 Feb 2009 07:33:23 -0800 PST
Subject: Duration'Image, Duration'Value and Duration'Last
Newsgroups: comp.lang.ada

[…]

I'm experimenting something strange when manipulating Duration'Image, Duration'Value and Duration'Last with GNAT :

The result of
Duration'Value(Duration'Image(Duration'Last))

is different from the result of

S : String := Duration'Image (Duration'Last)

and then

Duration'Value(S).

Here is a test program that shows the issue:

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Test is
   S : String   := Duration'Image
                     (Duration'Last);
   D : Duration := Duration'Value (S);
begin
   Put_Line (Duration'Image (D));
   Put_Line (Duration'Image
                     (Duration'Last));
end Test;
```

This test program gives me:

-9223372036.854775810

9223372036.854775810

Note that -9223372036.854775810 seems to be Duration'First...

Is it expected? Can someone tell me what happens here? Why this difference?

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 19 Feb 2009 07:44:01 -0800 PST
Newsgroups: comp.lang.ada
Subject: Re: Duration'Image, Duration'Value and Duration'Last

[…]

This is pretty obviously a GNAT bug. From experimentation, it appears that the problem is in the Duration'Value; Duration'Value ("9223372036.854775810") yields a negative value. Fixed-point types are usually represented as integers with an implied binary point somewhere in there. The integer whose leftmost bit is 1 and all other bits 0 is always tricky to deal with, because you can't negate it, so special care must be taken. If Duration'First is represented as exactly this integer, it's understandable why this sort of error may have come up.

From: Anonymous <anon@anon.org>
Date: Thu, 19 Feb 2009 17:00:16 GMT
Subject: Re: Duration'Image, Duration'Value and Duration'Last
Newsgroups: comp.lang.ada

It's actually a compiler bug. Even using a user-created type that contains the Duration range, will cause the same type of error.

```
type myDuration is
   delta 0.000000001
   range -((2 ** 63 - 1) * 0.000000001) ..
      +((2 ** 63 - 1) * 0.000000001);
   for myDuration'Small use
      0.000000001;
```

But if you manually use the "System.Val_Real" package which is the work horse for Duration'Value attribute, it does work correctly.

On Protected Objects

From: markp <markwork66@yahoo.com>
Date: Mon, 8 Dec 2008 05:34:36 -0800 PST
Subject: Quick Protected Object question
Newsgroups: comp.lang.ada

I have a very quick protected object question.

In a standard protected object setup as follows:

```
protected Test is
   procedure A;
   procedure B;
private
   Z : integer := 0;
end Test;

protected body Test is
   procedure A is
   begin
      < set of statements>
   end A;

   procedure B is
   begin
      < set of statements>
   end B;
end Test;
```

The question is this: when procedure A is called, are all threads that try to call B suspended until A finishes or, do the threads that call B execute as long as A is not touching the private data "Z". Is the lock at the procedure level or only at the data level?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 8 Dec 2008 05:43:37 -0800 PST
Subject: Re: Quick Protected Object question
Newsgroups: comp.lang.ada

[…]

At the procedure level; see ARM 9.5.1 at

http://www.adaic.com/standards/05rm/html/RM-9-5-1.html

From: micronian2@gmail.com
Date: Tue, 9 Dec 2008 11:59:54 -0800 PST
Subject: Re: Quick Protected Object question
Newsgroups: comp.lang.ada

[…]

In addition to the reference that Ludovic provided, here is a link that describes the Eggshell model that is used for protected objects:

http://www.iuma.ulpgc.es/users/jmiranda/gnat-rts/node25.htm

On the Activation of Ada Tasks

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 19 Feb 2009 09:37:31 -0800 PST
Subject: Language lawyer question: task activation
Newsgroups: comp.lang.ada

Should this program deadlock?

I don't think it should (and I think it should display "E1 accepted"), based on my understanding about when task activation is supposed to occur for the function result. But perhaps there's something about the relation between task activation and masters that I don't understand. Anyway, this hangs when I compile it with GNAT and run it - is this correct or not?

```
with Text_IO;
procedure Test is

   task type TType is
      entry E1;
   end TType;

   task body TType is
   begin
      accept E1 do
```

```
      Text_IO.Put_Line (
            "E1 accepted");
   end E1;
end TType;

function Func return TType is
begin
   return X : TType;
end Func;

procedure Do_It (X : TType) is
begin
   X.E1;
end Do_It;

begin
   Do_It (Func);
end Test;
```

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Fri, 20 Feb 2009 09:16:21 -0500
Subject: Re: Language lawyer question: task activation
Newsgroups: comp.lang.ada

[…]

> But isn't there another issue here: task types are limited, therefore Func is a constructor function, but in what object does it construct its returned value?

In the formal parameter X of Do_It.

From: Robert Matthews <ignored@ramatthews.free-online.co.uk>
Date: Fri, 20 Feb 2009 16:57:32 +0000
Subject: Re: Language lawyer question: task activation
Newsgroups: comp.lang.ada

[…]

So X from Do_It is passed as a hidden parameter to Func, so that X in Func is really whatever X in Do_It actually is - but what is that? I don't see an actual object of type TType in the example code. Calling Do_It with parameter Func seems a bit circular from this viewpoint.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Thu, 19 Feb 2009 18:54:50 -0500
Subject: Re: Language lawyer question: task activation
Newsgroups: comp.lang.ada

> Should this program deadlock? I don't think it should (and I think it should display "E1 accepted"), based on my understanding about when task activation is supposed to occur for the function result.

I think you're right. The task should be activated after Func returns, before calling Do_It.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 19 Feb 2009 18:57:10 +0100
Subject: Re: Language lawyer question: task activation
Newsgroups: comp.lang.ada

Cool! I am not a language lawyer but I think it is a bug. Even more funny it becomes with:

```
function Func return TType is
begin
  return X : TType do
    X.E1;
  -- Communicating with not yet
  -- returned object!
  end return;
end Func;
```

I guess that GNAT does not fire the task until its "construction," which happens too late in these cases. […]

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Thu, 19 Feb 2009 18:57:55 -0500
Subject: Re: Language lawyer question: task activation
Newsgroups: comp.lang.ada

```
> function Func return TType is
  begin
     return X : TType do
       X.E1;
      -- Communicating with not
      -- yet returned object!
     end return;
   end Func;
```

This one, however, should deadlock. The task is not activated until after Func returns. Func never returns, because it is waiting on an entry call of a not-yet-activated task.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 20 Feb 2009 14:22:12 +0100
Subject: Re: Language lawyer question: task activation
Newsgroups: comp.lang.ada

[…]

I.e. the behaviour of X declared of being TType depends on where it is declared. That's great. Let us consider this:

```
function Func return TType is
  function Func_Func return TType is
  begin
    return X : TType;
  end Func_Func;

  Y : TType := Func_Func;
begin
  return Z : TType := Func_Func do
    Y.E1; -- This is not like X?
    Z.E1; -- An this?
  end return;
end Func;
```

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Mon, 23 Feb 2009 08:36:54 +0100
Subject: Re: Language lawyer question: task activation
Newsgroups: comp.lang.ada

> I.e. the behaviour of X declared of being TType depends on where it is declared. That's great.

[…]

As it has always been. You cannot rendezvous with a task until it's activated. A task is activated after the "begin" for the frame where it is declared. Therefore yes, behaviour depends on where the task is declared.

[…]

Y is a local object. Z is a name (like a hidden parameter) that references the object that's being created by the function call. That object is declared in the *caller*.

On String I/O

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Tue, 25 Nov 2008 21:52:02 -0800 PST
Subject: Weird string I/O problem
Newsgroups: comp.lang.ada

The following program misbehaves if the line

```
Get(A_Float); -- PROBLEMATIC LINE
```

is present; the call to Get_Line is made but there is no opportunity for the user to enter a string - the program continues (I suppose) as though the user entered a line terminator, causing the following output:

```
Enter a float: 12.3
Enter a string: Hello from Get_Line.
Your string was
It was 0 long.
```

However, if the problematic line is commented out, the following occurs:

```
Enter a float: Enter a string: Hello from
Get_Line.
bla bla
Your string was bla bla
It was 7 long.
```

Here is the program:

```
with Ada.Text_IO, Ada.Float_Text_IO,
Ada.Strings.Unbounded;
use Ada.Text_IO, Ada.Float_Text_IO,
Ada.Strings.Unbounded;

procedure temp1 is
   procedure Get_Line(
      An_Unbounded_String : out
      Unbounded_String) is
     Max_Length : Integer := 256;
     A_String : String(1 .. Max_Length);
     Length : Integer;
   begin
     Put_Line("Hello from Get_Line.");
```

```
      Get_Line(A_String, Length);
      An_Unbounded_String :=
         To_Unbounded_String(
               A_String(1 .. Length));
   end Get_Line;

   UBS : Unbounded_String;
   A_Float : Float;

begin
   Put("Enter a float: ");
   Get(A_Float); -- PROBLEMATIC
                 -- LINE
   Put("Enter a string: ");
   Get_Line(UBS);
   Put_Line("Your string was " &
            To_String(UBS));
   Put_Line("It was" &
         Length(UBS)'img & " long.");
end temp1;
```

[…] I am running GNAT 4.3 on OS X 10.4.

From: Cristhoph Grein <christoph.grein@eurocopter.com>
Date: Tue, 25 Nov 2008 23:24:19 -0800 PST
Subject: Re: Weird string I/O problem
Newsgroups: comp.lang.ada

> Put("Enter a float: ");

> Get(A_Float); -- PROBLEMATIC LINE

Here you enter "12.3<Return>"

Get consumes the number and leaves <Return> in the input stream.

> Put("Enter a string: ");

> Get_Line(UBS);

Here, Get_Line consumes the rest of the previous input up to the <Return>, in your case the rest is empty.

(If you had entered 12.3 xxx<Return>, the contents of UBS would be " xxx".)

So you have no opportunity to enter a further string.

What you need, is a Skip_Line after the Get(A_Float).

[the sentence should be "Get_Line consumes the rest of the previous input up to and including the <Return>" as clarified in a subsequent post by the same author —mp]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 26 Nov 2008 09:25:19 +0100
Subject: Re: Weird string I/O problem
Newsgroups: comp.lang.ada

And a little advise for text processing:

1. Never read data items directly from files. Do lines first, as strings. Then parse obtained strings.
2. Take care to remove trailing LF and CR at the line end. (You never know if the text file do obey UNIX or MS-DOS conventions.)
3. Always verify that the whole line was parsed.
4. Do not use Unbounded_String, it is just an unnecessary overhead. Ada 2005 has Get_Line returning String. (In high-integrity software use Ada 95's Get_Line, that with the line size limited).

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 26 Nov 2008 10:07:53 +0100
Subject: Re: Weird string I/O problem
Newsgroups: comp.lang.ada

> What you need, is a Skip_Line after the Get(A_Float).

If you want to make sure you read something from a fresh new line, you can use set_col(1).

It has the nice property that if the previous get consumed the end_of_line, it does nothing, and if it didn't, it skips everything up to and including the end_of_line.

On Ada.Text_IO

From: Thomas Locke <thomas@kenshi.dk>
Date: Mon, 16 Feb 2009 21:36:01 +0100
Subject: C getchar() functionality in Ada
Newsgroups: comp.lang.ada

[…] I've started redoing all the K&R C examples in Ada, and I've already hit a wall that I'm not able to get around.

I have this C program:

```
#include <stdio.h>
int main(void){

   int c;
   while((c = getchar()) != EOF){
      putchar(c);
   }
   return 0;
}
```

It's very simple, but still I fail to mimick it in Ada.

The C program happily accepts everything I throw at it, and it responds with the expected values, whereas my Ada version(s) either fails at linefeeds, throw End_Error exceptions at me, or spits out too many linefeeds!

My current Ada code looks like this:

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Ito is
   C : Character;
begin
   while not End_Of_File loop
      Get (Item => C);
      Put (Item => C);
      if End_Of_Line then
         New_Line;
      end if;
   end loop;
end Ito;
```

This version works with input from keyboard and when I pipe some data into it like this: $ echo "FooBar" | ito

It craps out on files ($ cat SomeFile | ito), where it ignores linefeeds and throws End_Error exceptions at me when the last character in the file is a linefeed.

I've tried with Get_Immediate, Look_Ahead, Get_Line and a mixture of Strings, Characters and Unbounded strings. I just can't make it work.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 17 Feb 2009 17:40:28 +0100
Subject: Re: C getchar() functionality in Ada
Newsgroups: comp.lang.ada

[…]

In Ada, an end of line is not a character. Full stop. We have operations that deal with end of lines: new_line, skip_line, and end_of_line.

Therefore, your program should look like:

```
with Ada.Text_IO;
use Ada.Text_IO;

procedure Ito is
  C : Character;
begin
   while not End_Of_File loop
      if End_Of_Line then
         Skip_Line;
         New_Line;
      else
         Get (Item => C);
         Put (Item => C);
      end if;
   end loop;
end Ito;
```

From: Jeffrey R. Carter <spam.jrcarter.not@spam.acm.org>
Date: Tue, 17 Feb 2009 20:38:56 GMT
Subject: Re: C getchar() functionality in Ada
Newsgroups: comp.lang.ada

Except that, given a file that ends with a blank line, Ada.Text_IO.End_Of_File may return True before the final blank line has been detected. The general rule is to not use Ada.Text_IO.End_Of_File, and handle the resulting End_Error.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 18 Feb 2009 08:46:46 +0100
Subject: Re: C getchar() functionality in Ada
Newsgroups: comp.lang.ada

I agree. Moreover, it behaves better in the case of ill-formed files (from an Ada

point of view), such as those that have no terminating LF.

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Wed, 18 Feb 2009 02:41:40 -0800 PST
Subject: Re: C getchar() functionality in Ada
Newsgroups: comp.lang.ada

> I agree. Moreover, it behaves better in the case of ill-formed files (from an Ada point of view), such as those that have no terminating LF.

This includes what Jeffrey calls a blank line at end. Such a file is not well-formed in the Ada sense. This is discussed in Text_IO RM A. 10.2(3) and A.10.5(16).

From: Thomas Locke <thomas@kenshi.dk>
Date: Tue, 17 Feb 2009 20:46:56 +0100
Subject: Re: C getchar() functionality in Ada
Newsgroups: comp.lang.ada

[…]

> In Ada, an end of line is not a character. Full stop. We have operations that deal with end of lines: new_line, skip_line, and end_of_line.

> Therefore, your program should look like: […]

This program *almost* work exactly as the C version, but not quite. It fails to recognize all EOL's in files, and it behaves a bit "odd" when feeding data to it using the keyboard.

But I understand what you're saying: No end of line characters in Ada!

Except perhaps when using Get_Immediate and Look_Ahead?

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 18 Feb 2009 08:43:52 +0100
Subject: Re: C getchar() functionality in Ada
Newsgroups: comp.lang.ada

[…]

Depends on your definition of "odd". The time when you see the output may not be what you expect in relation to your input, because Text_IO needs some look-ahead to deal with page breaks; however, the output should correspond to your input (with maybe a small shift in time).

Remember too that the model is a file, and the keyboard is a device. It is in general acceptable to treat it as a file, but for things like timing, it is not.

> But I understand what you're saying: No end of line characters in Ada!
>
> Except perhaps when using Get_Immediate and Look_Ahead?

Look_Ahead has a parameter to tell when you are at end of line. The behaviour of Get_Immediate is less clear from the RM.

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Wed, 18 Feb 2009 03:42:49 -0800 PST
Subject: Re: C getchar() functionality in Ada
Newsgroups: comp.lang.ada

Get_Immediate falls a bit out of Text_IO, since it does not update the column, line and page counts.

It returns the next character in the file. If EoL is coded as characters, it will return them, else it will silently skip to the next line and return the first character. It simply returns any Control characters that might be in the file, other than the Get routines.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Newsgroups: comp.lang.ada

Subject: Re: C getchar() functionality in Ada
Date: Tue, 17 Feb 2009 18:44:02 -0500
> The Ada design has the same error here as Pascal did.

...Ada.Text_IO design...

Text_IO is a bit broken, I agree. But you can also use streams, or you can interface directly to the underlying OS primitives.

Ada has a lot of strengths, but unfortunately, the design of the Ada.Text_IO package is not one of them.

From: Jeffrey R. Carter <jrcarter@acm.org>
Newsgroups: comp.lang.ada

Subject: Re: C getchar() functionality in Ada
Date: Mon, 16 Feb 2009 21:43:23 GMT
> I have this C program:

> […]

> It's very simple, but still I fail to mimick it in Ada.

This program duplicates standard input on standard output.

You are being mislead by the use of "char" in C. "char" sometimes refers to the equivalent of Ada's Character, and other times to the equivalent of System.Storage_Elements.Storage_Element or Ada.Streams.Stream_Element.

This is one of the latter cases. You should not be using Character, String, or Ada.Text_IO for this. I suggest you look at streams and Ada.Text_IO.Text_Streams.

Networking Support in Ada

From: Oliver Kowalke <oliver.kowalke@gmx.de>
Date: Wed, 17 Dec 2008 19:33:32 +0100
Subject: networking support?
Newsgroups: comp.lang.ada

[…] Does Ada support IPsec (key management sockets)?

How is io-demultiplexing used (select, epoll, kqueue, /dev/pool,...) used?

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Thu, 18 Dec 2008 10:28:08 +0100
Subject: Re: networking support?
Newsgroups: comp.lang.ada

[…] These being OS functions, you probably have interfacing packages for your compiler. For some, you might be able to use POSIX packages; others, such as access to /dev/anything are really OS specific, not typically built into any programming language, but rather available as library calls.

[…]

From: Oliver Kowalke <oliver.kowalke@gmx.de>
Date: Thu, 18 Dec 2008 12:55:35 +0100
Subject: Re: networking support?
Newsgroups: comp.lang.ada

[…]

Should I implement functionality (for instance networking) in C++ classes and call them from Ada or use the C-functions (system calls) and implement the classes/behaviour in Ada (get benefits from Ada's safety)?

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Thu, 18 Dec 2008 13:47:17 +0100
Subject: Re: networking support?
Newsgroups: comp.lang.ada

[…]

How would your program benefit from first wrapping system functions (that use C conventions) in C++ classes only for having to match Ada's O-O types and C++'s O-O types in a second step?

Seems like an artificial setup to me.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Fri, 19 Dec 2008 13:49:27 -0800 PST
Subject: Re: networking support?
Newsgroups: comp.lang.ada

[…]

Except that a lot of things in the C world (especially in the system APIs) are defined in header files as preprocessor constants, optional or even unofficial structure fields or field order within a struct - all this means that the binary layout of data structures is a big question mark.

This problem is particularly notorious with network services.

There is no way to reasonably interface with all this mess from Ada and having a wrapper layer that fixes some of this optional/unofficial/ ordering stuff for the Ada part is a valid option.

Now the question is how thick this adapter layer should be - and here a layer that does a bit of lifetime management or error reporting in addition to just parameter passing is also a valid solution. Then you might discover that all the work that is done in this adapting layer can be

reused in C++ projects as well (it is a low-hanging fruit, so why not benefit from it?) and you end up with a regular C++ library that encapsulates some system services and is used from the Ada program.

From: Oliver Kowalke <oliver.kowalke@gmx.de>
Date: Sat, 20 Dec 2008 00:05:39 +0100
Subject: Re: networking support?
Newsgroups: comp.lang.ada

[…] my intention was to call some boost libraries (www.boost.org).

I'm not sure if C++-templates can be accessed/used by Ada.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Sat, 20 Dec 2008 11:23:54 -0800 PST
Subject: Re: networking support?
Newsgroups: comp.lang.ada

A significant part of the Boost libraries are template-based libraries entirely defined in their header files. There is even no object code to link to until you instantiate the target templates. Unfortunately there is no way to directly use them from Ada.
Which Boost libraries are you interested in?

From: Maciej Sobczak <maciej@msobczak.com>
Date: Sun, 21 Dec 2008 14:21:02 -0800 PST
Subject: Re: networking support?
Newsgroups: comp.lang.ada

[…]

> DateTime

> Filesystem

> System

> Interprocess

> (Regex)

I think that for these libraries your best option is to write simple wrappers with extern "C" interface (that is, C wrappers) and pragma Import that interface to Ada.

Note also that a significant part (if not all) of this is already available in Ada in the standard library or in the GNAT library - it might be easier and more natural to use what Ada offers first before reaching out to C++ libraries.

I would go for interfacing to an existing C++ library only when something much more specialized is involved. Numeric computation or communication infrastructure are possible examples.

A database library would be another example, see SOCI-Ada linked below.

[…]

[www.inspirel.com/soci-ada —mp]

From: Jacob Sparre Andersen <sparre@nbi.dk>
Date: 18 Dec 2008 10:54:29 +0100
Subject: Re: networking support?
Newsgroups: comp.lang.ada

> does Ada support IPsec (key management sockets)?

Yes and no.

ISO/IEC 8652:2007 does not mention IPsec, so in a sense it isn't supported directly in Ada.

But you can write programs using IPsec in Ada.

[…]

From: Anonymous
Date: Thu, 18 Dec 2008 16:14:21 GMT
Subject: Re: networking support?
Newsgroups: comp.lang.ada

If you are using GNAT then you can use "GNAT.Sockets" for High-level network programming and "GNAT.Sockets.Thin" for the low-level OS calls like select routine.

As for other routines, you must use an "pragma import" statement, because Ada does not directly support these calls.

[…]

Ada Code Generation from Simulink Models

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 20 Jan 2009 21:05:49 +0100
Subject: Ada code from Simulink models
Newsgroups: comp.lang.ada

Does anybody has any experience of generating Ada code from a MATLAB/Simulink model without MATLAB/Simulink Real-Time Workshop?

From: Jérôme Hugues <hugues@tarsis.enst.fr>
Date: Wed, 21 Jan 2009 08:25:57 +0000 UTC
Subject: Re: Ada code from Simulink models
Newsgroups: comp.lang.ada

I've heard of Beacon for Simulink, it generates Ada code from Simulink model, see:

http://www.adi.com/products_be_bss.htm

[…]

From: Colin Paul Gloster <Colin_Paul_Gloster@acm.org>
Date: Fri, 23 Jan 2009 08:36:22 +0000 UTC
Subject: Re: Ada code from Simulink models
Newsgroups: comp.lang.ada

I had been told that MatrixX was very good (though it was not actually MATLAB/Simulink) and I have never used it.

ANNA — A Language for Annotating Ada Programs

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Sat, 14 Feb 2009 15:19:52 -0800 PST
Subject: ANNA - A Language for Annotating Ada Programs
Newsgroups: comp.lang.ada

I've just learned about ANNA, “ANNA - A Language for Annotating Ada Programs”, but did not find online reference about it (may be it's too old).

Is it still alive? Do someone know about? Is it a kind of SPARK competitor ?

From: Marc A. Criley <mc@mckae.com>
Date: Sun, 15 Feb 2009 08:21:23 -0600
Subject: Re: ANNA - A Language for Annotating Ada Programs
Newsgroups: comp.lang.ada

It is quite old. I first encountered it at an Ada conference in 1984.

There's not much on the web, but if you have access to the “ACM Portal” library you can see what's available there with this very long query, will likely wrap:

http://portal.acm.org/results.cfm?query=PrimarySubject%3A"ANNA"&querydisp=PrimarySubject%2FNoun%3A"ANNA"&termshow=matchboolean&coll=GUIDE&dl=GUIDE

Overriding in Private Part

From: Maxim Reznik <reznikmm@gmail.com>
Date: Thu, 2 Oct 2008 08:49:10 -0700 PDT
Subject: overriding in private part
Newsgroups: comp.lang.ada

[…]

There is a package hierarchy of three packages: A1, A1.A2 and A1.A3.

And type hierarchy: A1.Base, A2.Child derived from A1.Base, A3.Child derived from A2.Child.

A1.Base has primitive procedure P2 in private part of A1.

A2.Child override P2 with its own P2 in private part of A2.

A3.Child is expected to override P2 with its own P2 in private part of A3, but actually it can't!

As result A3.P2 is never called in this example.

[…]

My expectation about this code, that A3.P2 override A1.P2 because private part of A1 is visible at this place, and A2.P2 invisible here, so have no influence here. But A2.P2 hides A1.P2 indeed.

Every type in class A1.Base'Class has P2 primitive subprogram, but any child of A2.Child can't override it, because P2 is hidden by A2.P2. It seems unnatural to me.

Errors of such kind are very difficult to find.

Is there any way to prevent such errors? (Besides keyword *overriding*)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 2 Oct 2008 18:17:28 -0500
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

This happened to us frequently when we were building Claw. That's the reason I pushed so hard to add what became the keywords to the language. It's an error that cannot be avoided or detected in Ada 95 (short of avoiding any declarations in private parts, which is nasty). It is very hard to predict what really is going to happen, so the keywords allow telling the compiler what you meant (and then it will complain if it disagrees).

[…]

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 3 Oct 2008 10:52:52 +0200
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

> In general, the language doesn't want to take private declarations of the parent into account when deciding whether something is overriding.

This is OK, but the problem is that the operation is formally considered private when it is obviously (in common sense) not.

> Your case is a little more confusing because, like the above example, there's an invisible P2 operation of the parent type that shouldn't affect the behavior, but this is inherited from a P2 operation of the *grandparent* type that *is* visible in the private part of A1.A3. Perhaps the compiler should generate a warning about the possible confusion in this case. If we didn't have an "overriding" keyword, I might even go so far as to suggest the language should make this case illegal, in order to prevent this kind of case from coming up.

It is not the overriding which must be illegal. The opposite should. I mean it should be:

overriding procedure P2
(X : Child); *-- Legal (presently illegal)*
not overriding procedure P2
(X : Child); *-- Illegal (presently legal)*

> But since "overriding" has been added, you should just use it, and it will solve the problem.

Unfortunately it does not. The programmer desired to override P2, and this is impossible to do, because there is a type in between which effectively *hides* the operation in all packages, regardless their visibility. Further, this behavior changes when the offending type is derived in A1 rather than in A1.A2. In that case A1.A3 would be able to override P2.

This is obviously broken to me.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Fri, 03 Oct 2008 16:29:21 -0400
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

> I shouldn't have said "solve". It *partially* solves the problem, in that if the programmers use "overriding" and "not overriding" consistently on everything, it won't let you write a program that has an unexpected result […]

I think you should say "overriding" wherever it's legal, but never say "not overriding". And use a compiler that warns on missing "overriding", such as a recent GNAT with the -gnatyO switch.

I agree that in the OP's example, P2 should be overriding - this is a language design flaw. But at least you won't get in trouble at run time, if you follow the above convention.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Sat, 04 Oct 2008 15:47:04 -0400
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

> I agree with Adam; after all, there is a reason that "not overriding" exists. I'm curious as to why you think it shouldn't be used. If you write a routine that you do not expect to override something, and it does anyway, you have a problem (because you could be called from a dispatching routine with a completely different purpose, and conceivably different preconditions and postconditions). I think you'd like to know about that problem.

If you use a compiler that warns on missing "overriding", then you don't need to say "not overriding", because that's the default - any subprogram that doesn't say "overriding" is not overriding.

Saying "not overriding" is just noise.

There is no good reason for "not overriding" to exist. If you "write a routine that you do not expect to override something, and it does anyway," then I don't agree you have a problem -- you get a warning.

The warning is crucial, of course. Without that, IMHO these indicators are nearly useless. If I were designing the language from scratch, I would make missing "overriding" an error. The only reason not to do that is for compatibility with Ada 95.

It's like parameter modes - you should never explicitly say "in", because that's the default. You should explicitly say "in out" or "out". It's unfortunate that "in" is allowed, because now the language has split into three dialects - those that say "in", those that never say "in", and those that say "in" for procedures but not functions.

That's not doing anyone any favors - the folks who write in those three dialects can't understand each other's code.

> There are cases where you can't use either indicator (where the overriding happens "late"), but those usually indicate a program structuring issue (routines are being hidden in the root types that ought to be visible, or the child types are making routines visible that should be private). And in such cases I think it is important to document why no indicator was given with a comment.

And I think there are some generics-related cases, too. But these are all corner cases. By and large, you can say "overriding" when you mean it, and let "no indicator" implicitly mean "no overriding".

> I wanted indicators to be allowed on all subprograms in order that I could enforce a rule of "no indicator" -> style violation, but that got voted down (they're only allowed on primitive subprograms). I still think that was a serious mistake (it's weird to have to leave out indicators on class-wide routines, for instance).

I agree that if you want an indicator on every subprogram declaration, then forbidding "no overriding" on some non-overriding subprograms is a language design mistake. But I don't want that - I want a safe (non-overriding) default.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Sun, 05 Oct 2008 15:57:48 -0400
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

> But non-overriding is unsafe. Taking your example with in and in-out, when the programmer uses in instead of in-out, that does not change the program semantics, so long the program remains legal. Otherwise (if the body actually changes the parameter) it will not compile. This is safe.

Good point, but I'm only half convinced. If you get in the habit of always saying "overriding" when appropriate, and you use the warnings, then you're unlikely to get into trouble. And the idea of putting "not overriding" all over the place seems awfully verbose, to me.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 6 Oct 2008 10:50:04 +0200
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

Both are unsafe. A felt verbosity comes from the strange decision to put [not] overriding in front of the declaration. If it were:

procedure Foo (X : Boo) is not overriding;

it would not be so offending.

As for me, I think that overriding could be a good default for all subprograms with at least one controlling argument. Declarations of any new primitive operation should then be explicit:

procedure Foo (X : Boo) is [abstract] new;

Non-primitive operations should be made illegal if any of the arguments is controlling:

```
type T is tagged ...;
procedure Foo (X : T) is new;

package Bar is
   procedure Foo (X : T); -- You cannot
                          -- do this!
   procedure Baz (X : T); -- Neither this!
   procedure Baz (X : T'Class); -- This is
                                -- OK
end Bar;
```

Maybe, that could be relaxed the bodies of the packages which specifications declare the type:

```
package A is
   type X is tagged ...;
end A;

package body A is
   procedure Some_Private_
                Stuff_Without_
                Redispatch (X : T)
      is not new;
end A;
```

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 6 Oct 2008 18:32:06 -0500
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

Yes, that would be the correct semantics if the language was being built from scratch. Unfortunately, the Ada 95 team decided to use the existing inheritance mechanism, and that isn't quite right. It's not so obvious until you try examples in various ways.

In any case, non-primitive routines with specific tagged types are pretty suspicious and clearly deserve a warning. (There are a couple of them in Claw - mostly functions returning some specific tagged type, but the majority are class-wide.)

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Sun, 05 Oct 2008 16:08:16 -0400
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

> "If you use a compiler that [does the right thing]" (or rather, if you use that compiler and the right set compiler switches) that works more or less fine. Except that you get only a warning on something that ought to be treated as an error. (Yes, I know that GNAT has a switch to treat warnings as errors.)

Shrug. To me, warnings and errors are pretty-much the same. At AdaCore, we compile everything in warnings-as-errors mode. Not always, but we have procedures in place that ensure no Ada code can escape into customer's hands with warnings.

> But a compiler-agnostic way to enforce the proper behaviour (overriding indicators for all the subprograms which actually override another subprogram) would be preferable, IMHO. I wish there was an Ada 2005 "pragma Overriding_Indicators_For_All_Overriding_Subprograms" or the like. Or did I overlook something like that in the standard?

I don't think you overlooked anything. I agree portability is nice, so it would nice to have a portable way to say this. Pragma Require_Overriding_Indicators might be a reasonable name.

But of course if you're worried about forgetting the warning switch, you should be equally worried about forgetting that pragma. Neither one seems like a big problem - it's something you do once, when setting up your project-wide pragmas, or project-wide build scripts. Much more likely to forget "overriding" when declaring a procedure, which is something you do every day.

Note that if you're writing code for several Ada compilers, you only need one of them to give the warning.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 6 Oct 2008 18:39:24 -0500
Subject: Re: overriding in private part
Newsgroups: comp.lang.ada

[…] errors required by the language are always more powerful than something done by a specific implementation (unless you can be sure that you are never going to use another implementation!).

Besides, I don't see how your warning would work with Ada 95 code (such as Claw). There's little value to having warnings in such code (it isn't yours anyway), but you still would want the errors in your own code. So it seems likely that you would have to turn the warning off in large amounts of code; that is going to make it more likely that it is also omitted in code where you want it to be checked.

> I don't think you overlooked anything. I agree portability is nice, so it would nice to have a portable way to say this. Pragma Require_Overriding_Indicators might be a reasonable name.

We couldn't figure out how such a pragma would work with generics. That's the main reason this is a three-state switch: "overriding", "not overriding", and nothing (which is essentially "don't care"). This may be a case where "perfect" prevented "good enough" - I surely wanted such a pragma, but it would have to have enough holes that it wasn't clear that it was worth anything.

On Tagged Types

From: Maciej Sobczak <maciej@msobczak.com>
Date: Fri, 28 Nov 2008 02:01:09 -0800 PST
Subject: Abusing tagged types
Newsgroups: comp.lang.ada

Is it considered to be a good practice to make a given type tagged only to benefit from the Object.Operation notation in Ada 2005?

Let's say there is a library where there are several types. Some of them are inherently tagged due to their design and they allow Obj.Oper notation out of the box. For others there is no design motivation to make them tagged (no dispatching calls for them, no Controlled, etc.) and as a result the whole library does not "feel" consistent, since Obj.Oper is not available across all exposed types.

The type can be made tagged *only* to get the syntax sugar.

Is it considered to be an abuse of the language feature?

From: Samuel Tardieu <sam@rfc1149.net>
Date: Fri, 28 Nov 2008 11:50:18 +0100
Subject: Re: Abusing tagged types
Newsgroups: comp.lang.ada

[…]

This will increase the memory size of every instance. Even if no dispatching is used the tag will be stored in each object because dispatching *could* be used.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Fri, 28 Nov 2008 05:28:38 -0800 PST
Subject: Re: Abusing tagged types
Newsgroups: comp.lang.ada

[…]

Let's say that in this case it does not matter anyway.

I'm asking about the design principles - the *purpose* of tagged type is to achieve polymorphism with dispatching calls. Is it OK to use tagged type without this motivation?

(well, Controlled is also in this category...)

By the way, what is the rationale for allowing Obj.Operation only for tagged types and not for all types? "Ada 2005 Rationale" does not seem to explain this.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 28 Nov 2008 06:08:41 -0800 PST
Subject: Re: Abusing tagged types
Newsgroups: comp.lang.ada

> By the way - what is the rationale for allowing Obj.Operation only for tagged types and not for all types? "Ada 2005

Rationale" does not seem to explain this.

Actually it does but I find the explanation a bit cryptic for untagged types other than access types:

"Other variations on the rules for the use of the notation were considered. One was that the mechanism should apply to untagged types as well but this was rejected on the grounds that it might add to rather than reduce confusion in some cases. In any event, untagged types do not have class wide types so they are intrinsically simpler.

It would have been particularly confusing to permit the notation to apply to access types especially an access type A referring to a tagged type T. If the access type and the tagged type both had the same or similar operations Op then ambiguities or errors could easily arise."

Maybe the AI has more details.

From: Adam Beneschan <adam@irvine.com>
Date: Mon, 1 Dec 2008 11:54:18 -0800 PST
Subject: Re: Abusing tagged types
Newsgroups: comp.lang.ada

[…]

From the beginning of AI-252: "Note: We considered generalizing this to allow non-tagged types to use this shorthand, but this becomes complex when the type is an access type, since both the access type itself, and its designated type must be considered. Furthermore, the benefit is lower since there is no issue of class-wide operations for non-tagged types, so all the "interesting" operations are all from a single package."

I don't see much other discussion in the AI about this issue; perhaps all the discussion about it was done elsewhere before AI-252 was opened.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 1 Dec 2008 22:04:57 -0600
Subject: Re: Abusing tagged types
Newsgroups: comp.lang.ada

[…]

I think most of the discussion on this came at the ARG meetings. (The text in the AI was the one-paragraph summary of the meeting conclusions, written by someone [me] that had lots of other things to do.) My recollection was that there was a problem with dealing with the automatic dereference and 'Access rules when the type could be an access type. It seemed that only one (having those rules) or the other (having access types) worked, and we choose the more useful semantics by limiting it to only tagged types.

Perhaps there is more detail in the meeting minutes (or maybe it is just lost).

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 28 Nov 2008 15:35:43 +0100
Subject: Re: Abusing tagged types
Newsgroups: comp.lang.ada

> I'm asking about the design principles - the *purpose* of tagged type is to achieve polymorphism with dispatching calls. Is it OK to use tagged type without this motivation?

One of the nice things in Ada is the ability to make a difference between a method and a subprogram (and yes, I'm afraid about the new generation of programmers who know only classes as unit of modularization, and call every subprogram a method).

IMHO, the term method should be reserved to operations that have a general semantics, but whose implementation (the way of doing the thing - the /method/ for doing the thing) depends on the type to which it applies. Methods make sense only in the presence of dynamic dispatching, i.e. the ability to tell an object: "do this your own way".

O.M notation stresses this by making clearer that a method is applied to a given object. Although subprograms may have a parameter of a non-tagged type which is in some sense distinguished, they are not methods by my definition, and I would not favour using that notation (to be honest, I've never been a great fan of the O.M notation in any case - it is misleading if you have more than one controlling operand, which is a great superiority of Ada over other OO languages).

Problem with Interfaces

From: Robert_Matthews <ignored@ramatthews.free-online.co.uk>
Date: Mon, 16 Feb 2009 09:53:59 +0000
Subject: Another problem with "interface"
Newsgroups: comp.lang.ada

In using interface types with GNAT I have encountered another problem.

Consider the following package:

```
package Test is

   type A_Type is limited interface;

   procedure P (A : in out A_Type;
      D : Integer) is abstract;

   protected type New_A_Type is
      new A_Type
   with
     procedure P (D : Integer);
     -- other subprograms...
   private
     F : Integer;
   end New_A_Type;

   function Set_A return New_A_Type;
end Test;
```

GNAT gives an error for the function Set_A:

"operation can be dispatching in only one type" […]

The version of GNAT is GNAT GPL 2008 (20080521).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 16 Feb 2009 14:29:50 +0100
Subject: Re: Another problem with "interface"
Newsgroups: comp.lang.ada

[…]

This looks like a compiler bug to me. The compiler thinks that Set_A is a protected function of New_A_Type and thus has the hidden argument New_A_Type and the result New_A_Type, so it complains. But protected type is not tagged so Set_A cannot be dispatching.

However you can trick the compiler by ensuring that Set_A declaration were beyond the freezing point of New_A_Type (whatever that might mean for a protected type). For example:

```
type A_Type is limited interface;

procedure P (A : in out A_Type;
    D : Integer) is abstract;

protected type New_A_Type is
    new A_Type with
  procedure P (D : Integer);
  -- other subprograms…
private
  F : Integer;
end New_A_Type;

package Foo is  -- Package brackets
                -- around it
  function Set_A return New_A_Type;
end Foo;
```

A more logical way to do it would be:

```
function Set_A return
   New_A_Type'Class;
```

Alas, this does not work, because there seem to be no way to create New_A_Type'Class, since New_A_Type is protected.

However you could use A_Type'Class instead:

```
function Set_A return A_Type'Class;
```

P.S. Returning New_A_Type from a function is tricky because it is limited, yet does not have aggregates. You can use the return statement to work this around:

```
function Set_A return New_A_Type is
begin
  return Result : New_A_Type do
```

```
    null;
  end return;
end Set_A;
```

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Mon, 16 Feb 2009 14:56:11 +0100
Subject: Re: Another problem with "interface"
Newsgroups: comp.lang.ada

[…]

> This looks like a compiler bug to me. The compiler thinks that Set_A is a protected function of New_A_Type and thus has the hidden argument New_A_Type and the result New_A_Type, so it complains. But protected type is not tagged so Set_A cannot be dispatching.

Maybe illustrating this point,

```
procedure R is
  type I is Synchronized Interface;

  protected type T is new I with
    function Make_T return I'class;
  end T;

  protected body T is
    function Make_T return I'class is
    begin
      return (I with null record);
    end;
  end T;

  X: constant T := T.Make_T;
begin
  null;
end;
```

```
=======================
GNAT BUG DETECTED
========================
| 4.3.0 20070903 (experimental) [trunk
revision 128061]
(i686-apple-darwin8) |
| Assert_Failure atree.adb:3812
|
| Error detected at r.adb:15:22
|
| Please submit a bug report; see
http://gcc.gnu.org/bugs.html.
```

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Mon, 16 Feb 2009 02:26:59 -0800 PST
Subject: Re: Another problem with "interface"
Newsgroups: comp.lang.ada

I guess you've confused GNAT beyond repair.

The problem is your function Set_A (the rest is OK). Ada 2005 no longer has return-by-reference functions, so you cannot return an object of type New_A_Type.

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Mon, 16 Feb 2009 11:40:01 +0100
Subject: Re: Another problem with "interface"
Newsgroups: comp.lang.ada

[…]

> The problem is your function Set_A (the rest is OK). Ada 2005 no longer has return-by-reference functions, so you cannot return an object of type New_A_Type.

I think that, still, you can call functions that will assign the (already existing) return object. […]

```
  X: constant New_A_T := Set_A;
```

http://www.adacore.com/2007/05/28/gem-3/

From: Robert_Matthews <ignored@ramatthews.free-online.co.uk>
Date: Mon, 16 Feb 2009 11:27:57 +0000
Subject: Re: Another problem with "interface"
Newsgroups: comp.lang.ada

[…] Indeed, that is my intent.

Note that if I use an ordinary record type, e.g.

```
type New_A_Type is new A_Type with
...
```

then GNAT compiles things OK; it is when I use a protected type it complains.

From: Egil Høvik <egilhovik@hotmail.com>
Date: Mon, 16 Feb 2009 03:45:42 -0800 PST
Subject: Re: Another problem with "interface"
Newsgroups: comp.lang.ada

Actually, it is when the protected type inherits an interface that it complains.

For me, at least, (using GNAT Pro 6.1.2), removing the inheritance compiles OK.

Suppression of Warnings and Pascal Ranges

From: Michael Mounteney <gate02@landcroft.co.uk>
Date: Mon, 29 Dec 2008 19:13:44 -0800 PST
Subject: Selective suppression of warnings -- gnat on GNU/Linux
Newsgroups: comp.lang.ada

[…] I am trying to build an application of which some of the source is automatically translated from Pascal, on the fly. The problem is that the automatically-translated source is causing a lot of spurious warnings about declarations not being used. This is because the Pascal code has many instances of:

```
type
  somerange = 1..10;
  somestruct = record ... end;
```

which is translated into Ada as

```
type somerange is new integer range
  1 .. 10;
type somestruct is record ... end
  record;
```

but the problem is that any operator such as + and = is not visible in other units. The solution to that is to rename the operators in the client units, thus:

```
with stage3;    -- contains definitions of
                -- somerange, somestruct etc.
package body myusage is
  function "=" (L, R :
      in stage3.somestruct)
    return Boolean
    renames stage3."=";

  function "+" (L, R :
      in stage3.somerange)
    return  stage3.somerange
    renames stage3."+";
      .....
end myusage;
```

without those renamings, any usage of = and + within the body of myusage are flagged as errors owing to lack of visibility/qualification.

The translator is a rather crude line-by-line affair written in Haskell that only performs partial analysis of the source, and certainly isn't up to identifying the arguments to operators within expressions. Thus, it produces the renaming clauses if it encounters the type name is the source; e.g., if it sees somerange, it outputs all the renamings for somerange. However, the renamings usually are not required, so GNAT warns about them. Normally, this would not be a problem; one would simply remove the unneeded declaration from the source. I did try putting the declarations into another package and then "with" and "use" that, but then the warning changes to "no declarations used from the package".

I really really really don't like "use" anyway and prefer always to qualify imported names.

What I'd like is a pragma that switches-off and switches-on the warning over the specific range of lines containing the renamings, but no such seems to be available. I don't want to switch off the warning from the command line as that will suppress valid warnings.

[…]

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Wed, 31 Dec 2008 11:46:08 -0800 PST

Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

[…]

Google reveals this Ada Gem of the Week from AdaCore:

http://www.adacore.com/2007/11/19/ada-gem-18/

It shows how to turn off warnings for a particular range of code. […]

```
package body Warnings_Example is
  procedure Mumble (X : Integer) is
  begin
    null;
  end Mumble;
end Warnings_Example;
```

will cause GNAT to complain:

```
warnings_example.adb:5:22: warning:
formal parameter "X" is not
referenced.
```

But the following will compile cleanly:

```
package body Warnings_Example is

  pragma Warnings (Off,
            "formal parameter ""X"" is not
             referenced");
  procedure Mumble (X : Integer) is
  pragma Warnings (On,
            "formal parameter ""X"" is not
             referenced");
    -- X is ignored here [...]
  begin
    null;
  end Mumble;

end Warnings_Example;
```

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Tue, 30 Dec 2008 18:13:37 -0500
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

[…]

GNAT has a whole bunch of ways to suppress warnings. Look at the docs. You can suppress warnings in a range of code. You can suppress particular types of warnings. You can suppress all warnings (in a range of code, or globally). Pragmas and command-line options.

As someone said, you might want "use type", which makes operators directly visible, but nothing else. Or you might want to use subtypes of Integer.

Three alternative translations of Pascal's:

type T = A..B;

have been discussed in this thread:

1. **subtype** T **is** Integer **range** A..B;
2. **type** T **is new** Integer **range** A..B;
3. **type** T **is range** A..B;

Option 1 matches Pascal semantics most closely.

2 and 3 both might be better Ada style in some cases, but:

It's hard to tell from the Pascal code whether it's better or worse. Sometimes 1 is better. It depends on how many type conversions are needed, and analyzing that would require a fairly sophisticated translator, with global analysis of the Pascal program.

Option 3 is questionable, because of overflow semantics for intermediate results in expressions. In Pascal, if you say (X+Y)/2, it won't overflow if X+Y is in Integer, but not in A..B. Same is True in Ada for option 2, but not necessarily for option 3.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 31 Dec 2008 10:46:34 +0100
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

> 1. subtype T is Integer range A..B;
> 2. type T is new Integer range A..B;
> 3. type T is range A..B;
> […]
> Option 3 is questionable, because of overflow semantics for intermediate results in expressions. In Pascal, if you say (X+Y)/2, it won't overflow if X+Y is in Integer, but not in A..B. Same is True in Ada for option 2, but not necessarily for option 3. Oh no! Option 2 has exactly the same problem, you just hope that by forcing your type to have the same number of bits as Integer (a type you know nothing about), there will be enough room for your computations...

If you are worried about overflows (and you use only additions), the proper declarations are:

```
type Big_Enough is range A .. 2*B;
subtype T is Big_Enough range A .. B;
```

Of course, if you compute more than single additions, a real analysis has to be done to determine the bounds of Big_Enough.

By all means, please, let's get rid of Integer!

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Wed, 31 Dec 2008 14:49:02 -0500
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

[…]

Well, since we're talking about grossly incompatible changes, we might as well have:

```
type String_Index is range
    1..<implementation-defined>;
type String is array (String_Index
    range <>) of Character;
```

More generally, it would also be nice to have a way to say, "Give me a number that would be appropriate as the upper bound of an array whose component type is T". The number would be guaranteed to be big enough that you would get Storage_Error if you ever create an array that big.

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Tue, 30 Dec 2008 11:01:18 +0000
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

> […] I am trying to build an application of which some of the source is automatically translated from Pascal, on the fly. […]
> This is because the Pascal code has many instances of:
> type
> somerange = 1..10;
> somestruct = record ... end;
>
> which is translated into Ada as
>
> type somerange is new integer range 1 .. 10;
> type somestruct is record ... end record;

[…]

No, the problem is that the Pascal subrange type declarations have been wrongly translated. The Pascal declaration:

type somerange = 1..10;

Means, in Ada:

```
subtype somerange is Integer
   range 1..10;
```

Make this change and the Ada type compatibility problems will magically vanish.

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Tue, 30 Dec 2008 12:37:33 +0100
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

I can see this is formally true, but wouldn't you loose the valuable distinction that comes from different numeric types?

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Tue, 30 Dec 2008 12:05:51 +0000
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

The point is that in Pascal they are NOT different types, so if one wants to mirror

the semantics of the Pascal program - which is presumably the reason for translating it - one must do as I say.

Even writing afresh in Ada, it is very unlikely indeed that one would want every subrange to be a separate type.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 30 Dec 2008 00:03:52 -0800 PST
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

[…]

> What I'd like is a pragma that switches-off and switches-on the warning over the specific range of lines containing the renamings, but no such seems to be available. I don't want to switch off the warning from the command line as that will suppress valid warnings.

Have you tried replacing the renaming declarations with "use type somerange"? One such clause applies to all operators. You will get fewer warnings, as if a single operator is used then the "use type" clause does not cause a warning.

From: Michael Mounteney <gate02@landcroft.co.uk>
Date: Tue, 30 Dec 2008 14:49:50 -0800 PST
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

Thanks very much, Ludovic, that answer was just what I needed!

The warning count has decreased by about 80%.

Bill: I take your point but I am trying to preserve semantic information the conversion. So I do want the types to be distinct.

The Pascal source has been modified where necessary thus:

```
dest := (*ada:typeofdest*)(expr);
```

which the convertor sees as

```
dest := typeofdest(expr);
```

in order to maintain the distinction of types.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Tue, 30 Dec 2008 18:26:33 -0500
Subject: Re: Selective suppression of warnings --- gnat on GNU/Linux
Newsgroups: comp.lang.ada

[…]

Aha! So you're not exactly translating Pascal to Ada, you're translating Pascal with some interesting annotations (in Pascal comments) to Ada.

I think you might be better off annotating the Pascal type declarations (which ones should be separate types, in the Ada sense, versus subtypes), rather than annotating the individual Pascal expressions with conversion annotations.

Closures in Ada

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 16 Oct 2008 16:26:49 +0200
Subject: Re: Defining a binary operator between function access types: Is it possible?
Newsgroups: comp.lang.ada

> I'm trying to define a binary operator between function access types that returns the access to the function that is the binary operator acting on both functions.

You cannot create a new subprograms and return it from another subprogram.

To have the effect of an interpreted language you have to define corresponding first-class types. For example:

```
type Operation is abstract
   tagged private;
function Evaluate (
            F : Operation;
            X : Real) return Real
   is abstract;
…
type Squaring is new Operation
   with null record;
overriding function Evaluate (
            F : Squaring;
            X : Real) return Real;
Square : constant Squaring;
…
type Composed_By_Plus is
   new Operation with private;


function "+" (L, R : Operation'Class)
   return Composed_By_Plus;
overriding function Evaluate (
   F : Composed_By_Plus; X : Real)
return Real;
…
```

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Thu, 16 Oct 2008 18:18:59 -0400
Subject: Re: Defining a binary operator between function access types: Is it possible?
Newsgroups: comp.lang.ada

You can't do that sort of thing in Ada. What the OP is asking for is "full closures". Ada only has "downward closures" - you can pass procedures into procedures, but not out.

> …This isn't going to work at all.

> A Function_Access is basically a pointer to a function that has already been written. You're asking the program to build a *new* function on the fly and return a pointer to that. That pretty much works only in interpreted languages, and Ada isn't one of those.

Full closures are typically supported by functional languages, such as Lisp (which Adam mentioned), Scheme, SML, OCaml, Haskell, etc. They do not need to be "interpreted languages" - it is reasonable to compile to machine code in many such languages, and it's done in practice.

They do normally need garbage collection.

Jacob Sparre Andersen outlined how to simulate full closures using tagged and class-wide types.

[…]

From: Ivan Levashew <octagram@bluebottle.com>
Date: Mon, 20 Oct 2008 23:46:59 -0700 PDT
Subject: Re: Defining a binary operator between function access types: Is it possible?
Newsgroups: comp.lang.ada

Nobody seems to mention another way to do the trick. One can use just downwards closures, and circumvent upwards closures restriction via CPS, e. g. add extra "Continuation : access procedure" argument in every procedure, and add extra "Continuation : access procedure (Result : Your_Result_Type)" argument in every function.

It will actually work. Despite being clearly crazy (it blows your stack), blowing stack and heap is what actually happens when C++ compiler interprets BOOST, Loki, Blitz++ sources. IIUC standards-compliant C++ compiler can't get rid of any of intermediate results.

P.S.

http://okasaki.blogspot.com/2008/07/functional-programming-inada.html

Performance Quirk

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 6 Nov 2008 18:44:09 -0600
Subject: Re: Interesting performance quirk.
Newsgroups: comp.lang.ada

> Now the interesting part. My main development system is a Windows XP laptop. On this system my "optimized" Blowfish benchmark encrypts or decrypts at about 11 MB/s (curiously decryption is a little faster than encryption, which seems odd). It also happens that I have OpenSUSE 10.2 Linux running on the same box in a VMware virtual machine. In that environment my benchmark encrypts or decrypts at fully 27 MB/s. It's over twice as fast! I'm using GNAT GPL 2008 in both cases with the same compiler options and exactly the same source code. I'm even using the same basic hardware although, as I said, one

> of my systems---the faster one---is a virtual machine.
>
> Should I be surprised at this performance difference? I wasn't expecting it. Note that I'm using Ada.Calendar.Clock to track execution time. At first I wondered if the virtual machine's notion of time was distorted in some way but, no... the program is definitely faster in the VM (it runs long enough so that the difference is speed is easily perceptible by a human).

I can't answer whether you should be surprised, but I'm not. My experience is that modern CPU chips have performance characteristics that seem random and depend on things that no one has any control over.

My most recent example was a hobby program, much think yours. I was surprised to see that fixing a memory management flaw caused the program to run twice as fast. That temporarily caused rejoicing, until improving the behavior of a non critical piece of the program caused the program to slow by 50%! (This effect showed up on several Windows OSes on different Intel processors. But not on the old Pentium IIIs.) Experimenting, I discovered that I could change code in units totally unrelated to the "hot" areas of the program and cause vast changes in the performance of the inner loops.

I verified of course that the generated code really was unchanged (it was).

I went as far as reading the latest Intel literature on these topics (and it is huge). I thought that the effect might have had something to do with the alignment of the innermost loops, but adding options to control that to Janus/Ada didn't help much (it did get rid of the slowest versions, but the performance still could vary wildly, about 30% if I remember correctly).

Having wasted most of a nice weekend messing with this (and having no customer requirements at the time), I finally gave up and just twiddled with some unrelated code until the program ran fast.

So I don't quite know what is going on. I suspect it is related in some way to alignment, but it might be necessary for some code to be page aligned for maximum performance (and that is way too expensive to use within loops and other code that is going to be executed - you have to fill the empty space with no-ops, and executing them takes some time. Intel actually recommends no-op sequences to use to fill space in order to minimize time - yuck).

So it is possible that the performance difference has everything to do with unrelated parts of your program (such as the I/O libraries), which are going to be different for the two OSes. And nothing to do with your Ada code or anything that your compiler has control over.

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 *Conferences and events for the international Ada community* at: http://www.cs.kuleuven.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programs, URLs, etc. and are updated regularly.

2009

April 01-04 — 2nd IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2009), Denver, Colorado. Topics include: Verification & Validation, Quality Assurance, Empirical studies, Embedded and real-time software, Concurrent software, etc.

April 06-08 — 2nd **International Conference on Trusted Computing** (Trust'2009), Oxford, UK. Topics include: implementation technologies for trusted platforms; implementations of trusted computing; verification of trusted computing architectures; etc.

April 14-16 — 16th Annual IEEE **International Conference and Workshops on the Engineering of Computer Based Systems** (ECBS'2009), San Francisco, California, USA. Topics include: Component-Based System Design; Design Evolution; Distributed Systems Design; ECBS Infrastructure (Tools, Environments); Education & Training; Embedded Real-Time Software Systems; Formal Methods; Integration Engineering; Model-Based System Development; Modeling and Analysis of Complex Systems; Open Systems; Reengineering & Reuse; Reliability, Safety, Dependability, Security; Standards; Verification & Validation; etc.

April 20-23 — 21st Annual **Systems and Software Technology Conference** (SSTC'2009), Salt Lake City, Utah, USA.

☺ May 16-24 — 31st **International Conference on Software Engineering** (ICSE'2009), Vancouver, Canada. Topics include: Specification and Verification; Software Architecture and Design; Patterns and Frameworks; Reverse Engineering, Refactoring, and Evolution; Tools and Environments; Empirical Software Engineering; Development Paradigms and Software Processes; Component-based Software Engineering; Model Driven Engineering; Distributed Systems and Middleware; Embedded System; Open Standards and Certification; Software Economics; Dependability (safety, security, reliability); Case Studies and Experience Reports; etc. Deadline for early registration: April 11, 2009.

- ☺ May 18 — 2nd **International Workshop on Multicore Software Engineering** (IWMSE'2009). Topics include: Modeling techniques for multicore software; Software components and composition; Programming languages/models for multicore software; Compilers for parallelism; Testing and debugging parallel applications; Software reengineering for parallelism; Operating system support, scheduling; Development environments for multicore software; Experience reports from research or industrial projects; etc.

May 25-27 — 9th **International Conference on Computational Science** (ICCS'2009), Baton Rouge, Louisiana, USA. Theme: "Compute, Discover, Innovate".

- ☺ May 25 — 6th **International Workshop on aPpplications of declArative and object-oriented Parallel Programming** (PAPP'2009). Topics include: high-level parallel language design, implementation and optimisation; modular, object-oriented, functional, logic, constraint programming for parallel, distributed and grid computing systems; industrial uses of a high-level parallel language; etc.
- ☺ May 25 — **Workshop on Using Emerging Parallel Architectures for Computational Science**. Topics include: Languages, models, tools, and compilation techniques for emerging architectures; etc.

☺ May 25-29 — 23rd IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2009), Rome, Italy. Topics include: Parallel and distributed algorithms; Applications of parallel and distributed computing;

Parallel and distributed software, including parallel programming languages and compilers, runtime systems, fault tolerance, middleware, libraries, scalability, programming environments and tools, etc.

☺ May 26-29 **DAta Systems In Aerospace** (DASIA'2009), Istanbul, Turkey.

☺ June 02-04 14th IEEE **International Conference on the Engineering of Complex Computer Systems** (ICECCS'2009), Potsdam, Germany. Topics include: Avionics and Automobile Software; Formal Methods and Approaches to Manage and Control Complex Systems; Interoperability and Standardization; Real-time and Embedded Systems; Software Architecture and System Engineering; Systems and Software Safety and Security; Tools, Environments, and Languages for Complex Systems; Verification Techniques for Complex Software Systems; etc.

June 03-06 5th **International Conference on Open Source Systems** (OSS'2009), Skövde, Sweden. Topics include: Software engineering perspectives (F/OSS development environments; Testing, assuring and certifying F/OSS quality and security; F/OSS usability, scalability, maintainability and other quality issues; F/OSS and standards, ...); Emerging perspectives (Licensing, IPR and other legal issues in F/OSS; F/OSS and innovation; ...); Studies of F/OSS deployment (Case studies of F/OSS deployment, migration models, success and failure; F/OSS in vertical domains and the 'secondary' software sector, e.g., automotive, telecommunications, medical devices; F/OSS applications catalog; ...); etc.

♦ June 08-12 14th **International Conference on Reliable Software Technologies - Ada-Europe'2009**, Brest, France. Sponsored by Ada-Europe, in cooperation with ACM SIGAda.

June 09-12 4th IFIP **Conference on Distributed Computing Techniques** (DisCoTec'2009), Lisbon, Portugal. Includes the FMOODS/FORTE'2009, DAIS'2009, and Coordination'2009 conferences.

June 09-11 11th International **Conference on Languages, Models, and Architectures for Concurrent and Distributed Software** (Coordination'2009). Topics include: Distributed and Concurrent Programming Models (multicore programming, data parallel programming, event-driven programming, ...); Distributed Software Management (component and module systems for distributed software, configuration and deployment architectures, ...); Case Studies (application of novel distributed and concurrent techniques); etc.

June 09-11 IFIP **International Conference on Formal Techniques for Distributed Systems** (FMOODS/FORTE'2009). Formed joinlty from the 11th Formal Methods for Open Object-Based Distributed Systems (FMOODS) and the 29th Formal Techniques for Networked and Distributed Systems (FORTE). Topics include: Languages and Semantic Foundations (new modeling and language concepts for distribution and concurrency, semantics for different types of languages, including programming languages, modeling languages, and domain specific languages; real-time aspects; ...); Formal Methods and Techniques (design, specification, analysis, verification, validation and testing of various types of distributed systems); Practical Experience with Formal Methods (industrial applications, case studies and software tools for applying formal methods and description techniques to the development and analysis of real distributed systems); etc.

June 09-11 9th IFIP **International Conference on Distributed Applications and Interoperable Systems** (DAIS'2009). Topics include: Innovative distributed applications; Models and concepts supporting distributed applications; Middleware supporting distributed applications; Software engineering of distributed applications; etc.

June 15-21 ACM SIGPLAN **Conference on Programming Language Design and Implementation** (PLDI'2009), Dublin, Ireland. Topics include: the design, development, implementation, evaluation, and use of programming languages; including: Extracting parallelism from programs, Exploiting explicit parallelism in programs, Memory management, Language constructs for parallelism, Program analyses, Type systems and program logics, Debugging techniques and tools, Language designs and extensions, Checking or improving the safety, security, or correctness of programs, etc.

☺ June 19-20 ACM SIGPLAN/SIGBED **Conference on Languages, Compilers, and Tools for Embedded Systems** (LCTES'2009). Topics include: Programming language issues in embedded systems, including Language features to exploit multi-core, single-chip SIMD, ..., Language features for distributed real-time control, and other complex

embedded systems, Language features to enhance reliability and security, Virtual machines, concurrency, inter-processor synchronization mechanisms, memory management techniques; Compiler issues in embedded systems, including Interaction between embedded computer architectures, operating systems and compilers, Support for debugging, profiling, exception and interrupt handling, for reliability and security, etc.; Tools for analysis, specification, design and implementation of embedded systems, including Distributed real-time control, and other complex systems, Validation and verification, system integration and testing, Timing analysis, timing predictability, WCET analysis and real-time scheduling analysis, Performance monitoring and tuning, Runtime system support for embedded systems, etc.; Novel embedded architectures.

June 16-18 **Code Generation 2009**, Cambridge, UK. Topics include: Tool and technology development and adoption; Code Generation and Model Transformation tools and approaches; Defining and implementing modelling languages; Language evolution and modularization; etc.

☺ June 18 **Workshop on Entwicklung zuverlaessiger Software-Systeme**, Regensburg, Germany. Topics include (in German): Realzeitanforderungen, Sprachen, Nachweis der Erfuellung der Sicherheits-Anforderungen, etc.

June 22-24 4th International **Workshop on Systems Software Verification** (SSV'2009), Aachen, Germany. Theme: "Real Software, Real Problems, Real Solutions". Topics include: static analysis, model-driven development, embedded systems development, programming languages, verifying compilers, software certification, software tools, experience reports, etc.

June 22-26 29th **International Conference on Distributed Computing Systems** (ICDCS'2009), Montreal, Canada. Topics include: findings in any aspects of distributed and parallel computing, such as Distributed Middleware, Reliability and Dependability, Security, etc.

June 23-26 5th **European Conference on Model Driven Architecture Foundations and Applications** (ECMDA-FA'2009), Enschede, the Netherlands. Topics include: Metamodeling foundations and tools; Model Transformation and Code Geneation; MDA for Complex Systems and Systems of Systems; MDA for Embedded Systems and Real-Time Systems; MDA for High-Integrity Systems, Safety-Critical, and Security-Critical Systems; MDA in the Automotive, Aerospace, Telecommunications, Electronics Industries; Comparative Studies of MDA Methods and Tools; MDA for Legacy Systems; etc. Deadline for submissions: April 6th, 2009 (tools, posters).

June 26 – July 02 21st **International Conference on Computer Aided Verification** (CAV'2009), Grenoble, France. Topics include: Algorithms and tools for verifying models and implementations, Program analysis and software verification, Verification techniques for security, Applications and case studies, Verification in industrial practice, etc.

☺ June 26-27 **Workshop on Exploiting Concurrency Efficiently and Correctly** $(EC)^2$. Topics include: advances in programming languages and tools for developing concurrent software; programming constructs for concurrency; formalization of concurrency libraries; verification tools; introducing concurrency in education; etc.

☺ June 29 DSN2009 - **Workshop on Architecting Dependable Systems** (WADS'2009), Lisbon, Portugal. Topics include: the structuring, modelling, and analysis of dependable software systems, such as: Rigorous design (architectural description languages, formal development, ...); Verification & validation (theorem proving, type checking, ...); Fault tolerance; System evaluation; Enabling technologies; Application areas (safety-critical systems, embedded systems, ...); etc.

☺ June 29 – July 03 47th **International Conference Objects, Models, Components, Patterns** (TOOLS Europe'2009), Zurich, Switzerland. Topics include: all aspects of object technology and neighboring fields, in particular model-based development, component-based development, and patterns (design, analysis and other applications); more generally, any contribution addressing topics in advanced software technology; contributions showcasing applications along with a sound conceptual contribution are particularly welcome.

☺ June 29 – July 03 9th **International Conference on New Technologies of Distributed Systems** (NOTERE'2009), Montreal, Canada. Topics include: Middleware; Existing paradigms revisited: object, component, ...; Fault-Tolerance and dependability; Information assurance and security; Formal methods and tools; etc.

☺ July 01-03 21st **Euromicro Conference on Real-Time Systems** (ECRTS'2009), Dublin, Ireland. Topics include: applications (consumer electronics; multimedia and entertainment; process control; avionics, aerospace; automotive; telecommunications); software technologies (compiler support, component-based approaches, middleware and distribution technologies, programming languages, operating systems); system design and analysis (modelling and formal methods, reliability and security in RT systems, scheduling and schedulability analysis, worst-case execution time analysis, validation techniques, ...).

June 30 9th **International Workshop on Worst-Case Execution Time Analysis** (WCET'2009). Topics include: any issue related to timing analysis, in particular Integration of WCET and schedulability analysis; Evaluation, case studies, benchmarks; Tools for WCET analysis; Program design for timing predictability; Integration of WCET analysis in development processes; WCET analysis for multi-threaded and multi-core systems; etc. Deadline for submissions: April 13, 2009.

July 01-03 9th **International Conference on Application of Concurrency to System Design** (ACSD'2009), Augsburg, Germany. Topics include: (Industrial) case studies of general interest, gaming applications, consumer electronics and multimedia, automotive systems, (bio-)medical applications, internet and grid computing, ...; Synthesis and control of concurrent systems, (compositional) modelling and design, (modular) synthesis and analysis, distributed simulation and implementation, ...; etc.

July 03-08 14th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2009), Paris, France.

July 05-12 36th **International Colloquium on Automata, Languages and Programming** (ICALP'2009), Rhodes, Greece. Topics include: Parallel and Distributed Computing; Principles of Programming Languages; Formal Methods and Model Checking; Models of Concurrent and Distributed Systems; Models of Reactive Systems; Program Analysis and Transformation; Specification, Refinement and Verification; Type Systems and Theory; etc.

☺ July 06-10 23rd **European Conference on Object Oriented Programming** (ECOOP'2009), Genova, Italy. Topics include: research results or experience in all areas relevant to object technology, including work that takes inspiration from, or builds connections to, areas not commonly considered object-oriented; examples are: Analysis, design methods and design patterns; Concurrent, real-time or parallel systems; Distributed systems; Language design and implementation; Programming environments and tools; Type systems, formal methods; Compatibility, software evolution; Components, Modularity; etc. Deadline for submissions: May 1, 2009 (posters, demos). Deadline for early registration: May 20, 2009.

☺ July 06 19th **Doctoral Symposium and PhD Students Workshop**. Topics include: Design Patterns, Concurrency, Real-time, Embeddedness, Distribution, Language Workbenches, Generative Programming, Language Design, Language Constructs, Static Analysis, Language Implementation, Methodology, Practices, Design Languages, Software Evolution, Formal methods, Tools, Programming environments, etc.

☺ July 07 1st **International Workshop on Distributed Objects for the 21st Century** (DO21'2009). Topics include: State-of-the-art distributed object systems; Language abstractions for developing Software as a Service; Combining objects with other paradigms (e.g. events, publish/ subscribe, tuples, dataflow, REST, ...); Alternative (non-OO) approaches to the above (and their pros/cons); etc. Deadline for submissions: April 8, 2009.

☺ July 07 8th **Workshop on Parallel/High-Performance Object-Oriented Scientific Computing** (POOSC'2009). Topics include: identifying specific problems impeding greater acceptance and widespread use of object-oriented programming in scientific computing; proposed and implemented solutions to these problems; and new or novel approaches, techniques or idioms for scientific and/or parallel computing. Specific areas of interest include: alternatives or extensions to mainstream object-oriented languages (e.g. C++, Java); performance issues and their realized or proposed resolution; issues specific to handling or abstracting parallelism, including the handling or abstraction of heterogeneous and multicore microarchitectures; higher level languages (e.g. domain specific languages) or their embedding into OO languages to support parallelism or specific tasks in scientific computing; grand visions (of relevance); etc. Deadline for submissions: April 8, 2009.

July 13-16 2009 **International Conference on Software Engineering Theory and Practice** (SETP'2009), Orlando, Florida, USA. Topics include: Case studies, Component-based software engineering, Critical software engineering, Distributed and parallel software architectures, Education aspects of software engineering, Embedded software engineering, Model Driven Architecture (MDA), Model-oriented software engineering, Object-oriented methodologies, Program understanding, Programming languages, Quality issues, Real-time software engineering, Real-time software systems, Reliability, Reverse engineering, Software design patterns, Software maintenance, Software reuse, Software safety and reliability, Software security, Software specification, Software tools, Verification and validation of software, etc. Event includes: special session on Object-Oriented Programming.

☺ July 13-16 WORLDCOMP2009 - **International Conference on Parallel and Distributed Processing Techniques and Applications** (PDPTA'2009), Las Vegas, Nevada, USA. Topics include: Parallel/Distributed applications; Reliability and fault-tolerance; Real-time and embedded systems; Software tools and environments for parallel and distributed platforms: operating systems, compilers, languages, debuggers, monitoring tools, software engineering on parallel/distributed systems, ...; Object oriented technology and related issues; Scheduling and resource management; etc.

July 19-23 ACM **International Symposium on Software Testing and Analysis** (ISSTA'2009), Chicago, USA.

☺ July 19 2nd **International Workshop on Defects in Large Software Systems** (DEFECTS'2009). Topics include: Techniques to detect, locate, or predict defects; Empirical studies of defects; Types of defects that occur in software; Evolution of defects over time; Tools for post-deployment defect detection and reporting; Experience using certain techniques to identify or predict defects; etc. Deadline for submissions: April 13, 2009.

July 29-31 3rd IEEE **International Symposium on Theoretical Aspects of Software Engineering** (TASE'2009), Tianjin, China. Topics include: Specification and Verification; Program Analysis; Model-Driven Engineering; Software Architectures and Design; Object Orientation; Embedded and Real-Time Systems; Component-Based Software Engineering; Software Safety, Security and Reliability; Reverse Engineering and Software Maintenance; Type System; Dependable Concurrency; etc.

☺ August 10-12 7th IEEE **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2009), Chengdu and Jiuzhai Valley, China. Topics include: all aspects of parallel and distributed computing and networking, such as Parallel/distributed system architectures, Tools and environments for software development, Distributed systems and applications, Reliability, fault-tolerance, and security, etc.

August 10-13 28th Annual ACM SIGACT-SIGOPS **Symposium on Principles of Distributed Computing** (PODC'2009), Calgary, Alberta, Canada.

August 24-28 7th **Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering** (ESEC/FSE'2009), Amsterdam, the Netherlands. Topics include: Specification and verification, Software architecture and design, Tools and environments, Software quality and performance, Formal methods, Component-based software engineering, Distributed systems and middleware, Embedded and real-time systems, Open standards and certification, Dependability (safety, security, reliability), Case studies and experience reports, etc. Deadline for submissions: May 4, 2009 (doctoral symposium abstracts), June 5, 2009 (demos, posters).

☺ August 25-28 15th International **European Conference on Parallel and Distributed Computing** (Euro-Par'2009), Delft, the Netherlands. Topics include: all aspects of parallel and distributed computing, such Support tools and environments, High performance architectures and compilers, Distributed systems and algorithms, Parallel and distributed programming, Multicore and manycore programming, Theory and algorithms for parallel computation, etc.

☺ August 25 EuroPar2009 - 3rd **Workshop on Highly Parallel Processing on a Chip** (HPPC'2009). Topics include: programming models, languages and software libraries, implementation techniques, support and performance tools, performance evaluation, parallel algorithms and applications, migration of existing codebase, teaching of parallel computing, for/on highly parallel multi-core systems. Deadline for paper submissions: June 5, 2009.

☺ Aug 31 – Sep 04 10th **International Conference on Parallel Computing Technologies** (PaCT'2009), Novosibirsk, Russia. Topics include: New developments, applications, and trends in parallel computing technologies;

All aspects of the applications of parallel computer systems; Languages, environment and software tools supporting parallel processing; General architecture concepts; Teaching parallel processing; etc.

Aug 31 – Sep 05 20th **International Conference on Concurrency Theory** (CONCUR'2009), Bologna, Italy. Topics include: concurrency theory and its applications, e.g. semantics, cross-fertilization between industry and academia, etc. Deadline for submissions: April 10, 2009 (papers).

☺ September 01-04 **International Conference on Parallel Computing 2009** (ParCo'2009), Lyon, France. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments.

☺ September 01-04 4th **Latin-American Symposium on Dependable Computing** (LADC'2009), João Pessoa, Brazil. Topics include: Dependability of software (analysis, architecture, testing, verification & validation, software certification); Dependability of maintenance; Security; Dependability and human issues; Safety; etc. Deadline for submissions: June 15, 2009 (fast abstracts, student forum).

September 09-11 8th **International Conference on Software Methodologies, Tools, and Techniques** (SoMeT'2009), Prague, Czech Republic. Topics include: Software methodologies, and tools for robust, reliable, non-fragile software design; Automatic software generation versus reuse, and legacy systems, source code analysis and manipulation; Intelligent software systems design, and software evolution techniques; Software optimization and formal methods for software design; Software security tools and techniques, and related Software Engineering models; Software Engineering models, and formal techniques for software representation, software testing and validation; etc.

☺ September 12-16 18th **International Conference on Parallel Architectures and Compilation Techniques** (PACT'2009), Raleigh, North Carolina, USA. Topics include: Parallel computational models; Compilers and tools for parallel computer systems; Support for concurrency correctness in hardware and software; Parallel programming languages, algorithms and applications; Middleware and run-time system support for parallel computing; Reliability and fault tolerance for parallel systems; Modeling and simulation of parallel systems and applications; Parallel applications and experimental systems studies; etc.

September 14-17 Joint 8th **Working International Conference on Software Architecture and 3rd European Conference on Software Architecture** (WICSA/ECSA'2009), Cambridge, UK. Topics include: architecture description languages; architecture reengineering, discovery and recovery; software architects' roles and responsibilities, training, education and certification; etc. Deadline for submissions: April 10, 2009 (papers), April 20, 2009 (workshops), April 28, 2009 (tutorials).

September 16-18 12th **International Conference on Quality Engineering in Software Technology** (CONQUEST'2009), Nuremberg, Germany. Topics include: specific real-life case studies with detailed quality analysis and evaluation; quality engineering issues in domains such as Medical IT, Automotive, Avionics, Transport, and IT; etc.

☺ September 22-25 38th **International Conference on Parallel Processing** (ICPP'2009), Vienna, Austria. Topics include: Programming Models, Languages, and Compilers: from high-level abstractions to efficient code, etc.

October 04-09 ACM/IEEE 12th **International Conference on Model Driven Engineering Languages and Systems** (MoDELS'2009), Denver, Colorado, USA. Topics include: Development of domain-specific modeling languages, Tools and meta-tools for modeling languages and model-based development, Evolution of modeling languages and models, Experience stories in general (successful and unsuccessful), Issues related to current model-based engineering standards, Experience with model-based engineering tools, etc. Deadline for submissions: April 26, 2009 (abstracts), May 10, 2009 (papers).

October 05-06 2nd **International Conference on Software Language Engineering** (SLE'2009), Denver, Colorado, USA. Topics include: the engineering of artificial languages used in software development including general-purpose programming languages, domain-specific languages, modeling and meta-modeling languages, data models, and ontologies. Deadline for submissions: July 3, 2009 (abstracts), July 10, 2009 (papers).

♦ October 07-09 14th **International Real-Time Ada Workshop** (IRTAW'2009), Portovenere, Italy (tentative dates).

☺ October 12-14 IMCSIT2009 - 2nd **Workshop on Advances in Programming Languages** (WAPL'2009), Mragowo, Poland. Topics include: Compiling techniques; Domain-specific languages; Formal semantics and

syntax; Generative and generic programming; Languages and tools for trustworthy computing; Language concepts, design and implementation; Metamodeling and modeling languages; Model-driven engineering languages and systems; Practical experiences with programming languages; Program analysis, optimization and verification; Program generation and transformation; Programming tools and environments; Proof theory for programs; Specification languages; Type systems; etc. Deadline for submissions: May 5, 2009 (full papers).

☺ October 25-29 24th **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2009), Orlando, Florida, USA. Topics include: the intersection between programming languages and software engineering; key programming models and programming methods and related software engineering ideas, technologies, tools, and applications; critical evaluation of accepted practices, proposals for new programming models, exploration and extension of well-established models, and other novel approaches to building systems; etc. Deadline for submissions: July 2, 2009 (posters, demonstrations, student research competition, doctoral symposium, onward! films, student volunteers).

Oct 30 – Nov 07 16th **International Symposium on Formal Methods** (FM'2009), Eindhoven, the Netherlands. Theme: "Theory meets practice". Topics include: every aspect of the development and application of formal methods for the improvement of the current practice on system developments; of particular interest are papers on tools and industrial applications; etc. Deadline for submissions: December 22, 2009 (workshops), May 4, 2009 (papers).

♦ November 01-05 ACM Annual **International Conference on Ada and Related Technologies** (SIGAda'2009), Tampa Bay area, Florida, USA. Sponsored by ACM SIGAda, in cooperation with SIGCAS, SIGCSE, SIGPLAN, Ada-Europe, and Ada Resource Association. Deadline for submissions: June 30, 2009.

☺ November 02-03 14th **International ERCIM Workshop on Formal Methods for Industrial Critical Systems** (FMICS'2009), Eindhoven, the Netherlands. Topics include: Design, specification, code generation and testing based on formal methods; Verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability; Tools for the development of formal design descriptions; Case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; Impact of the adoption of formal methods on the development process and associated costs; Application of formal methods in standardization and industrial forums; etc. Deadline for submissions: April 7, 2009 (papers).

☺ December 09-11 15th IEEE **International Conference on Parallel and Distributed Systems** (ICPADS'2009), Shenzhen, China. Topics include: Parallel and Distributed Applications and Algorithms, Multi-core and Multithreaded Architectures, Resource Management and Scheduling, Security, Dependable and Trustworthy Computing and Systems, Real-Time Systems, etc. Deadline for submissions: June 1, 2009.

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

2010

☺ January 20-22 37th ACM SIGPLAN-SIGACT **Symposium on Principles of Programming Languages** (POPL'2010), Madrid, Spain. Topics include: all aspects of programming languages and systems, with emphasis on how principles underpin practice. Deadline for submissions: July 8, 2009 (abstracts), July 15, 2009 (papers).

March 20-25 **European Joint Conferences on Theory and Practice of Software** (ETAPS'2009), Paphos, Cyprus. Events include: FOSSACS, Foundations of Software Science and Computation Structures; FASE, Fundamental Approaches to Software Engineering; ESOP, European Symposium on Programming; CC, International Conference on Compiler Construction; TACAS, Tools and Algorithms for the Construction and Analysis of Systems.

Preliminary Call for Participation

14th International Conference on Reliable Software Technologies – Ada-Europe 2009

8-12 June 2009, Brest, France

http://www.ada-europe.org/conference2009.html

The Conference

The 14th International Conference on Reliable Software Technologies – Ada-Europe 2009 will take place in Brest, France, on 8-12 June 2009. The conference has established itself as an international forum for providers, practitioners and researchers into reliable software technologies. Following its consolidated tradition, the conference will span a full week, with at its centre from Tuesday to Thursday a three-day technical program accompanied by vendor exhibitions, and at either end on Monday and Friday a string of parallel tutorials and workshops. The previous editions in this conference series were held at Montreux, Switzerland ('96), London, UK ('97), Uppsala, Sweden ('98), Santander, Spain ('99), Potsdam, Germany ('00), Leuven, Belgium ('01), Vienna, Austria ('02), Toulouse, France ('03), Palma de Mallorca, Spain ('04), York, UK ('05), Porto, Portugal ('06), Geneva, Switzerland ('07) and Venice, Italy ('08).

The conference presentations will illustrate current work in the theory and practice of the analysis, design, programming, verification and maintenance of long-lived, high-integrity software systems for a variety of application domains. The program also features outstanding keynotes and a robust track of industrial presentations. As usual, the conference provides ample time for Q&A sessions, panel discussions and social events. Participants include practitioners and researchers from industry, academia and government organizations interested in the promotion and development of reliable software technologies.

Overall Program

The conference program altogether features 10 tutorials, a technical program of 19 thoroughly refereed papers, a collection of 6 industrial presentations reflecting current practice and challenges in real-life software projects, three eminent invited speakers, a rich exhibition, two workshops on Software Vulnerabilities and on AADL, and an especially attractive social program. Springer will publish the proceedings of the regular program of the conference as Volume 5570 of the LNCS. The Ada User Journal will publish the proceedings of the other tracks of the program.

Keynote Addresses

Three eminent keynote speakers have been selected to open each day of the core conference program:

- John Benito (Blue Pilot Consulting, USA), a leading member of the international standardization and programming languages community, will deliver a talk entitled: "*ISO JTC 1/SC 22/WG 23 work on programming language vulnerabilities*".
- Pierre Sens (LIP6, Université Pierre et Marie Curie, Paris), a leading researcher in software technologies for distributed systems who presents us fault tolerance technologies in a talk entitled: "*Fault tolerance in large scale distributed systems*".
- Peter H. Feiler (SEI/CMU, USA), a worldwide expert in architecture modeling and verification, will discuss how AADL can be used for such a purpose in a talk entitled: "*Validation of safety-critical systems with AADL*".

Tutorial Program

The two days at either side of the core conference program include a rich selection of tutorials delivered by domain experts who will cover a variety of topics of interest to the conference community:

- *Building cross language applications using Ada,* Quentin Ochem (AdaCore, France).
- *An introduction to parallel and real-time programming with Ada,* John Mc Cormick (University of Northern Iowa, USA).
- *Software fault-tolerance,* Pat Rogers (AdaCore, USA).
- *Software measures for building dependable software systems,* William Bail (MITRE, USA).
- *Modeling for schedulability analysis with the UML profile for MARTE,* Julio Medina (Universidad de Cantabria, Spain), Huascar Espinoza (CEA-List, France).
- *SPARK - the libre language and toolset for high-assurance software,* Roderick Chapman (Praxis High Integrity Systems, UK).
- *Hard real-time and embedded systems programming,* Pat Rogers (AdaCore, USA).
- *Designing real-time, concurrent, and embedded software systems using UML and Ada,* Rob Pettit (The Aerospace Corporation, USA).
- *Object-oriented programming in Ada 2005,* Matthew Heaney (On2 Technologies, USA).
- *Execution time: analysis, verification, and optimization in reliable systems,* Ian Broster (Rapita Systems, UK).

Technical Program

The technical program of the conference includes 19 thoroughly peer-refereed papers on a wealth of subjects pertinent to the conference themes, from submissions coming from as many as 19 countries worldwide. The program also features a collection of 6 industrial presentations of challenges faced and solutions devised in real-life projects and a half-day vendor-presentation session.

Exhibition

The exhibition will open in the mid-morning break on Tuesday and run continuously until the end of the afternoon break on Thursday. Breaks will last one full hour to allow attendees comfortable time to visit the exhibition.

Conference Venue

Brest enjoys a wonderful location, on the shores of a very large bay, with the beauty of inland and coastal Brittany all around it. The town of Brest was destroyed during the second world war and (quickly) rebuilt in a modern style. The city is a centre of commerce, combining the cobbled streets and fortifications of the old port with all the attractions and facilities of the modern city. Anyone who loves ports will want to see the dock yards. Its 17th century castle was spared by the bombs and gives a good point of view of the harbour. The Brest castle now houses a museum and offices of the harbour authorities. The fine arts museum Musee des Beaux-arts is worth finding for it's collection of Pont-Aben school paintings. Brest is famous today for being the home of Oceanopolis - the largest aquarium in Europe. The Oceanpolis is not just an aquarium but a huge research centre and exhibition of sea life and all it's aspects, with huge tanks of fish and sea mammals.

Social Program

The social program of the conference will open with a welcome reception at Oceanopolis: Brittany's sea park by the Marina in Brest. The Sea Park is organized around 3 buildings which highlight the diversity and the habitat of the seas around the world: a temperate building which presents sea life around Brittany, the tropical building displays tropical and colorful fishes and sharks, the polar building is home of about 40 penguins. The 50 aquariums make it possible to observe the various animals and plants in their habitat and numerous activities (interactive games, posters, movies, ...) provide valuable information to discover the under-sea world. Ellidiss Technologies sponsors the reception at Oceanopolis.

The conference banquet will take place by the sea side, in the charming village of Porspoder, located 25 km North West of Brest: this area provides a spectacular landscape of rough cliffs and sandy coves, and impressive marine streams due to the collision of the Channel and the Atlantic waters. AdaCore sponsors the banquet.

In cooperation with SIGAda 

For the latest information on the conference consult: `http://www.ada-europe.org/conference2009.html`

14TH INTERNATIONAL REAL-TIME ADA WORKSHOP
IRTAW-14

7-9 October 2009
Portovenere
Italy
http://events.math.unipd.it/irtaw14/

CALL FOR PAPERS

For over 20 years the series of **International Real-Time Ada Workshop** meetings has provided a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions, and has attracted participation from key members of the research, user, and implementer communities worldwide. Recent IRTAW meetings have significantly contributed to the Ada 2005 standard, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar profile.

In keeping with this tradition, and in light of Ada 2005 implementations beginning to appear, and thought of post Ada 2005 language changes, the goals of **IRTAW-14** will be to:

- examine experiences in using Ada 2005 for the development of real-time systems and applications;
- report on or illustrate implementation approaches for the real-time features of Ada 2005;
- consider the added value of developing other real-time Ada profiles in addition to the Ravenscar profile;
- examine the implications to Ada of the growing use of multiprocessors in the development of real-time systems, particularly with regard to predictability, robustness, and other issues;
- examine and develop paradigms for using Ada 2005 for real-time distributed systems, taking into account robustness as well as hard, flexible and application-defined scheduling;
- consider the definition of specific patterns and libraries for real-time systems development in Ada;
- identify how Ada relates to the certification of safety-critical and/or security-critical real-time systems;
- review the status and contents of ISO reports related to real-time Ada and consider the interest of developing new secondary standards or extensions;
- examine the status of the Real-Time Specification for Java and other languages for real-time systems development, and consider user experience with current implementations and with issues of interoperability with Ada in embedded real-time systems;
- consider the lessons learned from industrial experience with Ada and the Ravenscar Profile in actual real-time projects;
- consider the language vulnerabilities of the Ravenscar and full language definitions.

Participation at **IRTAW-14** is by invitation following the submission of a position paper addressing one or more of the above topics or related real-time Ada issues. Alternatively, anyone wishing to receive an invitation, but for one reason or another is unable to produce a position paper, may send in a one-page position statement indicating their interests. Priority will, however, be given to those submitting papers.

Position papers should not exceed ten pages in typical IEEE conference layout, excluding code inserts. All accepted papers will appear, in their final form, in the Workshop Proceedings, which will be published as a special issue of *Ada Letters* (ACM Press). Selected papers will also appear in the Ada User Journal.

Please submit position papers, in PDF format, to the Program Chair by e-mail: **neil@cs.york.ac.uk**

Program Committee

Neil Audsley (Program Chair), Ben Brosgol, Alan Burns, Michael González Harbour, Stephen Michell, Javier Miranda, Luís Miguel Pinho, Juan Antonio de la Puente, Jorge Real, José Ruiz, Tullio Vardanega (Local Chair) and Andy Wellings.

Important Dates

Receipt of Position Paper: **8 May 2009**
Notification of Acceptance: **22 May 2009**
Final Copy of Paper: **16 September 2009**
Workshop Date: **7-9 October 2009**

Call for Technical Contributions – SIGAda 2009

ACM Annual International Conference
on Ada and Related Technologies:
Engineering Safe, Secure, and Reliable Software

Hilton St. Petersburg Bayfront Hotel
Tampa Bay, Florida, USA
November 1-5, 2009

Submission Deadline: June 30, 2009
Sponsored by ACM SIGAda
http://www.acm.org/sigada/conf/sigada2009

SUMMARY: Reliability, safety, and security are among the most critical requirements of contemporary software. The application of software engineering methods, tools, and languages all interrelate to affect how and whether these requirements are met.
Such software is in operation in many domains of application. Much has been accomplished in recent years, but much remains to be done. Our tools, methods, and languages must be continually refined; our management process must remain focused on the importance of reliability, safety, and security; our educational institutions must fully integrate these concerns into their curricula.

The conference will gather industrial and government experts, educators, software engineers, and researchers interested in developing, analyzing, and certifying reliable, safe, secure software. We are soliciting technical papers and experience reports with a focus on, or comparison with, Ada.

We are especially interested in experience in integrating these concepts into the instructional process at all levels.

POSSIBLE TOPICS INCLUDE BUT ARE NOT LIMITED TO:

- Transitioning to Ada 2005
- Challenges for developing reliable, safe, secure software
- Ada and SPARK in the classroom and student laboratory
- Language selection for highly reliable systems
- Mixed-language development
- Use of high reliability subsets or profiles such as MISRA C, Ravenscar, SPARK
- High-reliability standards and their issues
- Software process and quality metrics
- System of Systems
- Real-time networking/quality of service guarantees
- Analysis, testing, and validation
- Use of ASIS for new Ada tool development
- High-reliability development experience reports
- Static and dynamic analysis of code
- Integrating COTS software components
- System Architecture & Design
- Information Assurance
- Ada products certified against Common Criteria / Common Evaluation Methodology
- Distributed systems
- Use of new Ada 2005 features/capabilities
- Fault tolerance and recovery
- Performance analysis

KINDS OF TECHNICAL CONTRIBUTIONS:

TECHNICAL ARTICLES present significant results in research, practice, or education. Articles are typically 10-20 pages in length. These papers will be double-blind refereed and published in the Conference Proceedings and in ACM Ada Letters. The Proceedings will be entered into the widely-consulted ACM Digital Library accessible online to university campuses, ACM's 80,000 members, and the software community.

EXTENDED ABSTRACTS discuss current work for which early submission of a full paper may be premature. If your abstract is accepted, you will be expected to produce a full paper, which will appear in the proceedings. Extended abstracts will be double-blind refereed. In 5 pages or less, clearly state the work's contribution, its relationship with previous work by you and others (with bibliographic references), results to date, and future directions.

EXPERIENCE REPORTS present timely results on the application of Ada and related technologies. Submit a 1-2 page description of the project and the key points of interest of project experiences. Descriptions will be published in the final program or proceedings, but a paper will not be required.

PANEL SESSIONS gather a group of experts on a particular topic who present their views and then exchange views with each other and the audience. Panel proposals should be 1-2 pages in length, identifying the topic, coordinator, and potential panelists.

WORKSHOPS are focused work sessions, which provide a forum for knowledgeable professionals to explore issues, exchange views, and perhaps produce a report on a particular subject. A list of planned workshops and requirements for participation will be published in the Advance Program. Workshop proposals, up to 5 pages in length, will be selected by the Program Committee based on their applicability to the conference and potential for attracting participants.

TUTORIALS offer the flexibility to address a broad spectrum of topics relevant to Ada, and those enabling technologies which make the engineering of Ada applications more effective. Submissions will be evaluated based on relevance, suitability for presentation in tutorial format, and presenter's expertise. Tutorial proposals should include the expected level of experience of participants, an abstract or outline, the qualifications of the instructor(s), and the length of the tutorial (half-day or full-day). Tutorial presenters receive complimentary registration to the other tutorials and the conference.

HOW TO SUBMIT: Send contributions by **June 30, 2009**, in Word, PDF, or text format as follows**:**

Technical Articles, Extended Abstracts, Experience Reports, and Panel Session Proposals: Program Chair, Lt. Col. Jeff Boleng (Jeff.Boleng@usafa.edu)

Workshop Proposals: Workshops Chair, Bill Thomas (BThomas@mitre.org)

Tutorial Proposals: Tutorials Chair, Richard Riehle (RDRiehle@nps.edu)

FURTHER INFORMATION:

CONFERENCE GRANTS FOR EDUCATORS: The ACM SIGAda Conference Grants program is designed to help educators introduce, strengthen, and expand the use of Ada and related technologies in school, college, and university curricula. The Conference welcomes a grant application from anyone whose goals meet this description. The benefits include full conference registration with proceedings and registration costs for 2 days of conference tutorials/workshops. Partial travel funding is also available from AdaCore to faculty and students from GNAT Academic Program member institutions, which can be combined with conference grants. For more details visit the conference web site or contact Prof. Michael B. Feldman (mfeldman@gwu.edu).

OUTSTANDING STUDENT PAPER AWARD: An award will be given to the student author(s) of the paper selected by the program committee as the outstanding student contribution to the conference.

SPONSORS AND EXHIBITORS: Please contact Alok Srivastava (Alok.Srivastava@auatac.com) for information about becoming a sponsor and/or exhibitor at SIGAda 2009.

IMPORTANT INFORMATION FOR NON-US SUBMITTERS: International registrants should be particularly aware and careful about visa requirements, and should plan travel well in advance. Visit the conference website for detailed information pertaining to visas.

ANY QUESTIONS?:

Please submit your questions on the conference to the Conference Chair, Greg Gicca (gicca@adacore.com) or Local Arrangements Chair Currie Colket (colket@acm.org).

Thirty Years of the Ada User Journal

John Barnes

11 Albert Road, Caversham, Reading, RG4 7AN, UK; Tel: +44 118 947 4125; email: jgpb@jbinfo.demon.co.uk

Abstract

This nostalgic paper looks back over the evolution of the Ada User Journal and highlights some memorable and forgettable events.

Keywords: Journal, Ada.

1 Introduction

Luís Miguel Pinho sent me an email recently pointing out that the Ada User Journal was about to embark on its thirtieth volume and could I possibly write a brief survey of its background to celebrate this event. I said yes and this paper is the outcome.

It so happens that I do have (thanks to Alan Burns) every copy going back to Volume 1, Number 1 issued in March 1980. So I had plenty of stuff to dig around in.

It wasn't always called Ada User Journal. Indeed it was not originally a publication of Ada-Europe at all. It started out as Ada UK News. But since the numbering system has remained continuous it is clearly in spirit the same publication.

Here are some key dates and volume numbers showing the transition from Ada UK News to the Ada User Journal.

1980	Volumes 1 – 6	Ada UK News
1986	Volumes 7 – 14	Ada User
1994	Volumes 15 – ??	Ada User Journal

Volumes 1–18 were published by Ada UK. Volumes 19–22 were published jointly by Ada UK and Ada-Europe. Volumes 23 onwards were and continue to be published by Ada-Europe.

2 Ada UK News

The first issue was simply some stapled pages of A4. The editorial was written by Prof. Ian Pyle then Chair of Computer Science at the University of York and the founding Chairman of Ada Language UK Ltd. Ian said:

"Each issue of the newsletter will be edited by a guest editor. I have the pleasure of starting the ball rolling, and taking the opportunity to say something about Ada in general. To my mind, the most important fact is that Ada has elicited interest in both the industrial and academic worlds. Perhaps for the first time we can find ourselves pulling in the same direction to the benefit of us all. This is an opportunity for bridging the industrial/academic gulf which we must not lose."

This first issue included reports on the Test and Evaluation Workshop in cool Boston in October 1979 and the Ada Environment Workshop in sunny San Diego in November.

One remark from each is worth noting. On Ada generally, the report from Boston said that there was enormous commitment to the language and that no-one said "Ada is a wretched language, we cannot use it"; the tenor was "Ada is great but needs refining". From San Diego, Jean Sammett did a straw poll on two questions: Should there be subsets of Ada? (40% yes, 30% no); and Will there be subsets of Ada? (90% yes, 0% No).

The next issues had rotating editors and were undoubtedly printed locally. Thus issue 2 was printed by Ferranti Computer Systems. These issues all had plain green covers although in some cases the green was unpleasant. Issues 1 and 2 are shown in Figure 1.

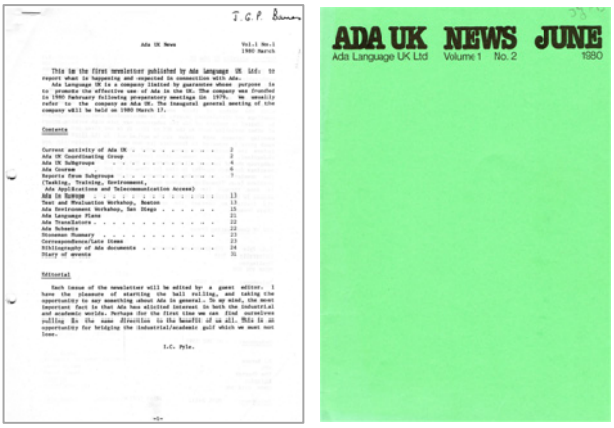

Figure 1 The first two issues of Ada UK News

Volume 2 was erratic and showed the typical problems of producing a regular publication: numbers 3 and 4 were rolled into one.

Volume 3 seems to have been even more erratic. There were only two issues and the first was bound with an ugly spiral binder.

But Volume 4 showed an enormous improvement. There was now a proper Publication Coordinator, Ros Spry, who made it all happen. And it now had a genuine ISSN number.

Those were exciting days. Ada was a focus for much thought. But she was slow both in coming and going. The compilers were slow to arrive and slow to go when they did arrive.

3 Ada User

Volume 7 in 1986 saw the introduction of a new name, Ada User. This aimed to reflect that it was not just an Ada UK publication but one for all those interested in Ada. Moreover, people were now actually using Ada and not just talking about it. So the journal had a new shiny and stripy green cover. And Ros Spry was now Publication Manager. In the editorial for the first of the new series, Brian Tooby of High Integrity Systems said:

"Some people may be surprised at how widespread Ada has become. It is now a truly international basis for technology, a common infrastructure for building software systems, a language with powerful host and target computers already designed around it."

Ian Pyle took over as permanent editor in July 1986 and the National Computing Centre in Manchester took over production in April 1987. This was partly because the NCC was now a Validation Authority for Ada.

1987 was also the year of the 1st International Real-Time Ada Workshop which was held in Devon and numbers 3 and 4 of Volume 8 contain interesting reports on the discussions.

Publication moved to Chapman and Hall in July 1990. And Dan Simpson of the then Brighton Polytechnic took over as editor from Volume 12 in January 1991. Publication with Chapman and Hall resulted in a much more professional appearance with material set in two columns. As Dan said in the editorial:

"... with this issue we are pleased to start our association with Chapman and Hall. They have an established reputation in both book and journal publishing and we are sure that the relationship ... will prove fruitful to everyone, particularly the readers. The improvements in format imply a slightly longer lead time in production but, as good software engineers, we shall improve the front-end of the life-cycle to keep the total elapsed time no longer than before."

But it was the time of the 1990s recession and Chapman and Hall gave up at the end of 1992. Production was transferred to IOS press in Amsterdam, and the format was reduced. Ada User seemed to be shrivelling away.

Figure 2 The first issues of Ada User and Ada User Journal

4 Ada User Journal

The name was changed to Ada User Journal with effect from Volume 15 in 1994. Curiously enough, there is no explanation of why in the editorial, and publication remained with IOS and in the reduced format. Figure 2 shows the first issues of Ada User and Ada User Journal.

But despite being smaller there was good stuff about Ada 9X to read about.

Starting with Volume 17, the A4 format was resumed and publication was brought back to the UK. Jim Briggs of Portsmouth became editor in June 1996

5 Ada-Europe News

Readers will recall that there have been two instantiations of Ada-Europe. The first Ada-Europe was run by the European Commission and was legally incorporated in Strasbourg. The first president of the old Ada-Europe was Garth Glynn. The second and current Ada-Europe is an independent Belgian company. The old Ada-Europe transferred its assets to the new Ada-Europe (with some difficulty but that is another story).

The new Ada-Europe started its own journal known as Ada-Europe News in June 1989. The first president was Toomas Kaer and he put together the first issue which like the first issue of Ada UK News was very plain.

Issues 2 to 6 were also plain but from issue 7 a really professional style was adopted as seen in Figure 3.

The editor of issues 2 to 8 was Juan António de la Puente and the editor of issues 9 to 11 was Francisco Gómez Molinero. They worked together as joint editors for issues 12-16. And then Albert Llamosi took over as editor from issue 17 in November 1993.

Ada Europe-News continued for several years as a high quality journal with valuable content and good presentation but it proved difficult to produce it on a regular basis.

In 1997, the boards of Ada UK and Ada-Europe reflected on the fact that there were three journals about Ada: Ada Letters in the US, Ada User Journal in the UK and Ada-Europe News in Europe.

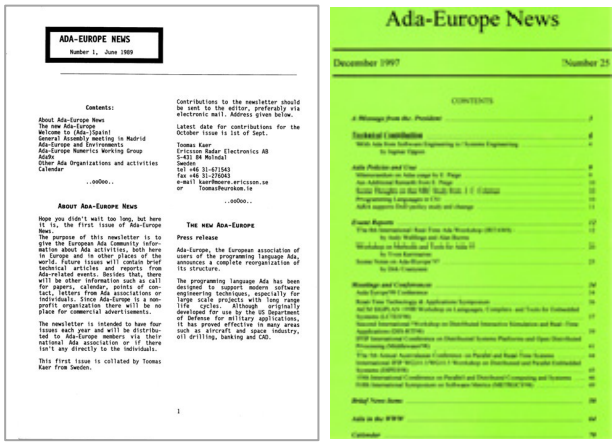

Figure 3 The first and last issues of Ada-Europe News

It was becoming clear that the community would be better served by a smaller number of better journals. Accordingly, Ada-Europe and Ada UK decided to merge Ada-Europe News and the Ada User Journal.

And so the last Ada-Europe News was that of December 1997. In the editorial, the then President remarked that it was an example of "The King is dead, Long Live the King!"

6 Ada User Journal revitalized

Volume 19 was the first of the new series with a greatly increased circulation and more material. Jim Briggs continued as editor and he was now assisted by Dirk Craeynest as news editor and Michael González Harbour as deputy editor.

It really was a new start with much more content, a newly designed cover and an enhanced editorial board.

It continued thus for three years with Jim Briggs as editor but Volume 22 brought yet another change as shown in Figure 4.

7 Professional quality

Although the journal was greatly improved with the merger with Ada-Europe News it still did not look quite as professional as it had been with Chapman and Hall ten years earlier. The problem was lack of uniform style and layout particularly with regard to submitted papers.

Accordingly, a complete redesign was decided upon. The accessibility of modern computer technology enabled authors to write everything for themselves so that we could bypass the need for typesetting with its risk of errors.

New glamorous colourful covers were designed under the guidance of Michael Gonzalez with the logos of Ada UK and Ada-Europe emblazoned on back and front. And templates were produced for articles in two columns and news in three columns to ensure uniformity of style.

Figure 4 Joint productions of Ada UK and Ada-Europe

And so was born the quality journal we have today. The first editor of the new journal was Neil Audsley of York. Ironically there was a minor mishap with the very first issue – the format was slightly incorrect with just a few millimetres trimmed from both dimensions.

8 Ada-Europe takes over

Ada UK ran into difficulties after one year of the new design and so Ada-Europe took over as the sole publisher from Volume 23 in 2002. And Tullio Vardanega became editor with Volume 23 number 3.

Tullio bravely remained editor for many years and handed over to our present editor Luís Miguel Pinho with effect from Volume 28 number 3.

Curiously, the very last issue edited by Tullio had difficulties – the front cover was not printed correctly and it was trimmed slightly too small.

This issue is shown in Figure 5. Note the Ada-Europe logo magnificently displayed on the back.

Figure 5 The issue of June 2007

Other matters I have not touched upon are printing and distribution – distribution is always a problem. Not only do issues have to be sent to members in many countries but for the sake of economy it is often the case that those for a particular country are sent in bulk and then redistributed by National Ada Organizations and noble volunteers.

And for historic reasons copies have to be sent to various copyright libraries such as those in Oxford, Cambridge and Dublin.

10 A pearl among journals

In marriages, 25 years is a silver event and 50 years is a golden one. But 30 years is the Pearl anniversary when you are supposed to give a Pearl to your spouse. So maybe we can find an appropriate acronym. As we know, Ada is a

> Precise Elegant And Reliable Language

Maybe readers can think of an improvement based perhaps on reserved words. Pragmatic, exceptional, and ??

Finally, let me finish by congratulating all those who have contributed material, time, and energy throughout the past thirty years both visibly as editors and authors and behind the scenes in production and distribution. So here's wishing many more years of publication to the Ada User Journal.

Progress Report: ISO/IEC 24772, Programming Language Vulnerabilities

James W. Moore

The MITRE Corporation, 7515 Colshire Drive, McLean, VA 22102, USA; email: James.W.Moore@ieee.org

John Benito

Blue Pilot Consulting, Inc, Santa Cruz, CA 95063-2998; USA; email: benito@bluepilot.com

Abstract

Any programming language has constructs that are imperfectly defined, implementation-dependent, or difficult to use correctly. As a result, software programs sometimes execute in a manner that is different than what was intended by the developer. In some cases, the unintended functionality can be exploited by hostile parties or can lead to failure when used in unanticipated circumstances. The result can be a compromise of safety, security, privacy, dependability or some other critical property. Security vulnerabilities are a particular concern because an adaptive adversary can use a compromise in any executing program—even a non-critical one—as a springboard to make additional attacks on other programs. This report describes an effort to develop an authoritative account of the known weaknesses in programming languages and how developers might avoid those weaknesses.

1 Introduction

Despite the fact that all programming languages have weaknesses, they manifest the weaknesses in different ways and the weaknesses must be mitigated in different ways, sometimes by better use of the language, sometimes by tooling such as static analysis, and sometimes by other methods such as review.

Some will be tempted to dismiss the problem, saying that one should simply use a better programming language. However, this viewpoint would overlook two factors:

- All programming languages have some weaknesses.
- The selection of a programming language for a project is often not a technical decision but is often forced by external concerns.

This article will describe progress on the planned ISO/IEC TR 24772, *Information Technology—Programming Languages—Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use*.

The project is being conducted in ISO/IEC JTC 1/SC 22/WG 23[1]. The WG has two officers—John Benito, convener and James Moore, secretary—the authors of this article.

2 The Technical Report

The "TR" in the designation of the document means that it is not a standard (a document that prescribes requirements for conformance), but a Technical Report—in this case, a Type 3 Technical Report—a document that provides guidance but not requirements. Therefore, the report will consist of information and recommendations. The report will describe programming language weaknesses in a generic manner that spans a broad selection of languages. However, since

- not all vulnerabilities are present in all languages;
- the ones that are present manifest themselves differently in different languages;
- and mitigation of the manifested vulnerabilities differ among the various languages

there is a need for language-specific material. Therefore, the report will include annexes that are specific to various programming languages. We plan to cooperate with other SC 22 working groups (the ones responsible for the standardized programming languages) to write these annexes. We also hope to obtain annexes for languages standardized by organizations that are outside of ISO/IEC.

Although the information in the Technical Report would be useful for the development of software required to exhibit any critically important property, the report is intended for four specific audiences:

- *Safety*: those developing, qualifying, or maintaining a system where it is critical to prevent behaviour that might lead to loss of human life or human injury, or damage to the environment.

[1] The International Electrotechnical Commission (IEC) develops standards for electrical and electronic devices; the International Organization for Standardization (ISO) develops standards for nearly everything else. They have a Joint Technical Committee (JTC 1) that deals with information technology. One of its subcommittees (SC 22) deals with programming languages. A working group (WG 23) of SC 22 is producing the subject document.

- *Security*: those developing, qualifying, or maintaining a system where it is critical to exhibit security properties of confidentiality, integrity, and availability.
- *Mission-Critical*: those developing, qualifying, or maintaining a system where it is critical to prevent behaviour that might lead to property loss or damage, or economic loss or damage.
- *Modeling and Simulation*: those who are primarily experts in areas other than programming but need to use computation as part of their work and who require high confidence in the applications they write and use.

The working group has taken two approaches to identifying weaknesses in programming languages. An *empirical* approach has relied on prior efforts that categorize particular classes of vulnerabilities that appear to occur frequently in the wild. This has been particularly helpful in finding security-related weaknesses because large numbers of security weaknesses result from a few identifiable patterns of attack, such as buffer overrun and execution of unvalidated remote content. An *analytical* approach has built on prior efforts that identified weaknesses via *a priori* analysis of particular programming languages. This has been particularly helpful in identifying safety-related weaknesses. We can speculate that it might also be helpful in identifying the security weaknesses of the future as current opportunities become less easily exploitable.

So the report will provide guidance to users of a broad range of programming languages. In some cases, a language-specific annex will provide specific guidance. However, the generic discussions will be useful for users of languages that are not specifically covered. The advice will assist users in improving the predictability of the execution of their software, even in the presence of an attacker or its use in anticipated circumstances. It will also inform their selection of an appropriate programming language for a project, when they have the freedom to make that choice.

The working group also plans an outcome in addition to the report itself. The working group will provide feedback to its sibling working groups, suggesting ways in which the standardized specification of the programming language might be improved so that predictability of execution would be improved.

The project has succeeded in gaining a broad base of participation. Measured in various ways we have participation from a variety of parties and interests. For example, at some level, we have participation from:

- Eight nations: Canada, France, Germany, Italy, Japan, Netherlands, United Kingdom, and USA
- Several programming languages: Ada, C, C++, C#, C++CLI, Cobol, Fortran, Java, MUMPS
- Some organizations with a strong interest in dependable software: the Computer Emergency Response Team (CERT) of Carnegie Mellon University, the US Food and Drug Administration, the US National Security Agency, and the Motor Industry Software Reliability Association

WG 23 has handled this project since its creation in September 2008; previously the work was performed by an ad hoc sub-group of SC 22 with the odd name of OWGV (standing for "other working group – vulnerabilities"). Like any ISO/IEC product, the report will go through a process of ever-widening consensus formation. Working Drafts were written by the working group and its predecessor. A Preliminary Draft Technical Report (PDTR) is currently under review and ballot by the parent, SC22. Comments from that ballot will be resolved and the ballot repeated as necessary until consensus is reached. Finally, the Draft Technical Report (DTR) will be balloted for approval by the grand-parent (JTC 1). Only after approval by at least 75% of the JTC 1 nations will the Technical Report be published—probably in early 2010. That will probably not be the end of the story—evolving attack patterns and the evolution of the language standards will require future revision of the report. Furthermore, there will be a continuing effort to "recruit" additional language-dependent annexes.

The working group conducts its work in person with nine meetings to date, supplemented by a wiki, an email reflector, and a website. The website can be accessed by the public at http://aitc.aitcnet.org/isai/.

The current body of the draft document contains seven major sections:

- Scope (an explanation of the intended use of the document)
- References (any other standards that one must use with this one)
- Terms and Definitions
- Symbols (none so far)
- Vulnerability Issues (an explanation of some general concepts)
- Programming Language Vulnerabilities (discussions of the programming language weaknesses)
- Application Vulnerabilities (other vulnerabilities that don't result from programming languages *per se* but which are related to programming language usage)

In the current draft, 48 programming language weaknesses are described as well as 18 application vulnerabilities.

The primary content of the Technical Report's body are the 48 descriptions of programming language weaknesses. All of them follow a uniform outline:

- Brief description of the vulnerability as it occurs in execution
- Cross-reference to enumerations and other classifications, e.g. CERT Coding Guidelines, Common Weakness Enumeration (CWE), Joint Strike Fighter (JSF) Coding Guidelines, MISRA C Coding Guidelines.
- Description of failure mechanism, i.e. how the coding problem leads to a vulnerability in the application

- Applicable language characteristics, i.e. the types of programming languages affected by the weakness
- Avoiding or mitigating the vulnerability, i.e. how one can code to avoid the problem or, in some other way, mitigate its effects
- Implications for standardization, i.e. recommendations for groups creating language standards

This format is best explained by an example from the current draft (Figure 1). Anything appearing in curly brackets {} is our explanation of the intended content rather than the content itself.

3 Conclusion

Currently, WG 23 has reasonably firm assurances that language-dependent annexes will be provided for Ada, C, and Fortran through cooperation with working groups 9, 14, and 5, respectively, of SC 22. We are hoping to find additional groups with the expertise to write annexes for other standards.

6.17 Boundary Beginning Violation [XYX] {Every description is assigned an arbitrary three-letter code. This allows one to reference a description even if subsequent versions of the report are reorganized.}

6.17.1 Description of application vulnerability

{This is intended to be a very brief description of the vulnerability as it occurs in execution.}

A buffer underwrite condition occurs when an array is indexed outside its lower bounds, or pointer arithmetic results in an access to storage that occurs before the beginning of the intended object.

6.17.2 Cross reference

{Cross references to CWE, JSF, MISRA, CERT, etc.}

6.17.3 Mechanism of failure

{This description is intended to depict the mechanism of failure connecting the programming language weakness to the vulnerability in the application.}

There are several kinds of failures (in some cases an exception may be raised if the accessed location is outside of some permitted range):

- A read access will return a value that has no relationship to the intended value, e.g., the value of another variable or uninitialized storage.
- An out-of-bounds read access may be used to obtain information that is intended to be confidential.
- A write access will not result in the intended value being updated and may result in the value of an unrelated object (that happens to exist at the given storage location) being modified.
- When the array has been allocated storage on the stack an out-of-bounds write access may modify internal runtime housekeeping information (e.g., a functions return address) which might change a program's control flow.

6.17.4 Applicable language characteristics

{If the report does not contain an annex providing information specific to the language of interest, the reader may consult this section to determine whether the language is likely to have this weakness.}

This vulnerability description is intended to be applicable to languages with the following characteristics:

- Languages that do not detect and prevent an array being accessed outside of its declared bounds.
- Languages that do not automatically allocate storage when accessing an array element for which storage has not already been allocated.

6.17.5 Avoiding the vulnerability or mitigating its effects

{This section describes, in generic terms, how the problem might be avoided or mitigated. A language-specific annex might contain more specific information.}

Software developers can avoid the vulnerability or mitigate its ill effects in the following ways:

- Use of implementation provided functionality to automatically check array element accesses and prevent out-of-bounds accesses.
- Use of static analysis to verify that all array accesses are within the permitted bounds. Such analysis may require that source code contain certain kinds of information, e.g., that the bounds of all declared arrays be explicitly specified, or that pre- and post-conditions be specified.
- Sanity checks could be performed on all calculated expressions used as an array index or for pointer arithmetic.

Some guideline documents recommend only using variables having an unsigned type when indexing an array, on the basis that an unsigned type can never be negative. This recommendation simply converts an indexing underflow to an indexing overflow because the value of the variable will wrap to a large positive value rather than a negative one. Also some languages support arrays whose lower bound is greater than zero, so an index can be positive and be less than the lower bound.

In the past the implementation of array bound checking has sometimes incurred what has been considered to be a high runtime overhead (often because unnecessary checks were performed). It is now practical for translators to perform sophisticated analysis that significantly reduces the runtime overhead (because runtime checks are only made when it cannot be shown statically that no bound violations can occur).

6.17.6 Implications for standardization

{This section suggests how language standards might be improved to improve the problem.}

- Languages that use pointer types should consider specifying a standard for a pointer type that would enable array bounds checking, if such a pointer is not already in the standard.

Figure 1 Report draft example

Distributed Status Monitoring and Control Using Remote Buffers and Ada 2005

Bradley J Moore

General Dynamics Canada, 1020 - 68th Ave N.E. Calgary, Alberta, Canada T2E 8P2; Tel: +011 403 730-1367; email: brad.moore@gdcanada.com

Abstract

The ability to monitor status and control equipment distributed over a network is a common network management need. This paper describes an approach to the design of a prototype dynamic network where the assets of vehicles on a network can be monitored and controlled at multiple remote stations. In particular, Ada 2005 features are explored in conjunction with Ada's Distributed Systems Annex (DSA) features to utilize a suite of remote buffer classes that implement an interface providing a mechanism for sharing a distributed dataset. In addition, the paper demonstrates an approach for distributed interoperability between Ada and C++ by using the DSA to distribute a C++ class hierarchy of objects that can be accessed by application code written in both languages. Finally, the paper exposes a need and describes a possible solution for enhancing existing DSA implementations in order to extend support for more complex and constrained networking environments.

Keywords: Ada 2005, Remote Interfaces, Multicast, Distributed Systems Annex, C++, MANET, DTN

Acknowledgements

The General Dynamics Canada IR&D team, Mark Adcock, Al Young, Curtis Osiowy, Jamel Mouallem for introducing the author to DTN technologies and concepts, and to Marius Ghinescu for providing funding assistance for travel to support the presentation of this paper, and to Thomas Orth, and Chris McPhee, for providing moral support, and to my dear wife Heather, and three children Alyson, Sarah, and Jackson for their support and understanding while the author worked on the paper and prototype. Their assistance is duly noted and appreciated.

1 Introduction

This paper suggests how Ada 2005 can be utilized to create a distributed system that allows a dynamic network of vehicles to be managed remotely from a number of monitoring/management stations. The system accounts for vehicles and monitoring stations joining and leaving the network, and allows the monitoring stations to select the classes of equipment to be managed. A wireless network

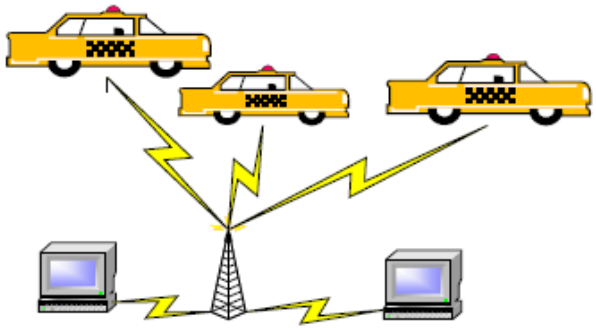

Figure 1: A semi-static MANET

constructed in such a manner is known as a Mobile Ad-hoc Network (MANET). Networking in a MANET environment has its own set of challenges. For instance, there may be an absence of central servers, and IP addresses of peers on the network may not be known. Further challenges arise if the networking environment involves interlinked heterogeneous networks where end to end connectivity between nodes cannot be guaranteed. Such a functioning network is known as a Delay Tolerant Network (DTN) [1]. This paper sketches out a design for a basic peer to peer MANET where nodes are connected on such a challenged network. The paper also draws attention to a perceived gap in availability of DTN networking solutions written in Ada. This may become an important niche area to complement Ada's existing niche in safety critical and high reliability systems.

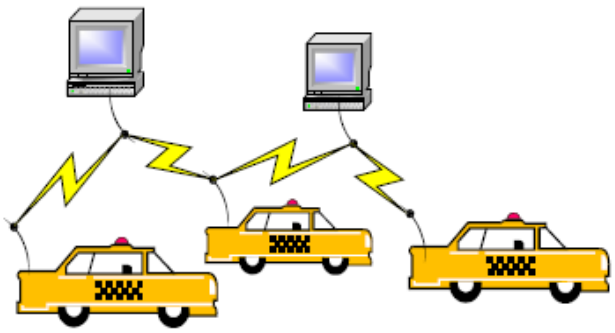

Figure 2: A peer to peer MANET

This paper is based in the industrial talk “Distributed Status Monitoring and Control using Remote Buffers and Ada 2005”, presented at Ada-Europe 2008, and in the paper of the same name presented at SIGAda 2008 and published in Ada Letters, XXVII (3), December 2008.

2 Application Prototype

A monitoring station may potentially be receiving numerous status updates from multiple vehicles at the same time. Buffering is used to accommodate such bursts in data traffic. In particular, a reusable generic remote buffer container abstraction is utilized allowing the monitoring application to run more independently from the networking communications. There exists one such buffer instance in every monitoring application, and vehicles are able to write data to the monitor application's buffer remotely.

There exists a hierarchical library of generic buffer implementations [2] that can be combined in various ways to assemble composite buffer implementations for different purposes.

An Ada 2005 limited interface defines the set of operations that can be performed on a buffer container, and multiple layers of generic instantiation are possible to allow a programmer to combine a low level buffer implemention (for example a Bounded Buffer, or an Unbounded Buffer) with a higher level synchronous implementation such as a Passive Buffer, or a Ravenscar Buffer, to provide the desired concurrency functionality.

Different monitoring applications might use different classes of buffers depending on the needs of the application. For example, an embedded application might choose to use a bounded buffer class which has no heap allocation, where the memory resources for the buffer are allocated statically before the execution begins.

A desktop application might choose to use an unbounded buffer which can grow in size to accommodate larger data bursts, where memory usage is not so critical. A safety critical device might choose a Ravenscar buffer, and a logging device might choose to use a persistent buffer to log to a file that is circular and bounded in size.

Another goal is to show interoperability with another language. The idea of using the DSA for data distribution in a large system may be more appealing if it can be done without restricting the choice of programming languages used throughout the system. Since the GNAT compiler knows how to interface with the Gnu C++ Application Binary Interface (ABI), the Ada compiler can directly interface with C++ objects from this compiler [3]. Also, since GCJ, the GNU Java compiler, shares the same ABI as Gnu C++, Ada should be able to interface with Java objects [4] for this compiler.

The C++ language does not directly support the streaming of Ada objects however, so the technique used involves deriving Ada classes from each concrete C++ class and distributing the Ada classes via DSA. C++ code is passed pointers to the distributed objects, which C++ sees as regular C++ class objects. One advantage of this approach is that the Ada compiler is able to generate the marshalling and unmarshalling code used to stream the C++ objects in most cases, since the derived classes are actually Ada objects. The programmer does not need to provide such routines.

Another advantage of this approach is that the details of configuring the communications can be left out of the application code. This provides a simpler abstraction for the programmer, since the programmer is able to operate on objects as if they were local objects without having to deal with meta-language code generators and third party communication libraries that may not be as portable to future target platforms.

A further advantage arises because the compiler is aware of the distributed nature of the library units, it can assist the programmer in the design to help ensure that troublesome constructs affecting the reliability of the distributed applications are avoided.

3 Design summary

A Vehicle application runs in every vehicle. Each vehicle maintains it's own status for all assets that are inside the vehicle. In this prototype system, there are currently four types of vehicle assets that can be monitored.

- A fuel sensor showing the amount of remaining fuel in a fuel tank.
- A GPS device indicating the vehicle location.
- A Camera that can be mounted in multiple places on the vehicle to capture the view from various angles as the vehicle moves around. A video capture file may be requested remotely, and transferred.
- A control device that can lock or unlock the doors of the vehicle. The doors may be locked or unlocked remotely from a monitoring station.

In this prototype, there are two classes of distributed applications that provide the functionality.

- Vehicle applications
- Monitoring Station applications

4 Vehicle design

Vehicles can arrive and leave the network at any time. Whenever a status item changes in the vehicle, an update is sent out (published) on the network which can be processed by any interested (subscribing) clients. Also, an interface is provided so that a client can request the current status snapshot for all assets of interest in a particular vehicle, which in turn are sent (as a response) directly to the client from the vehicle. A client can discover vehicles and their assets using a multicast variant of this call, or the client may make the request directly (unicast) to the vehicle if the client has the Remote Access to Class-Wide (RACW) value that designates the vehicle object.

An Update call is provided which allows the vehicle or a remote client to update the vehicle's set of status records. This is also a control feature, as the vehicle can respond to a remote update request by configuring the vehicle equipment to match the status request values. This could then trigger a multicast status update to be sent (published) from the vehicle to interested (subscribing) clients.

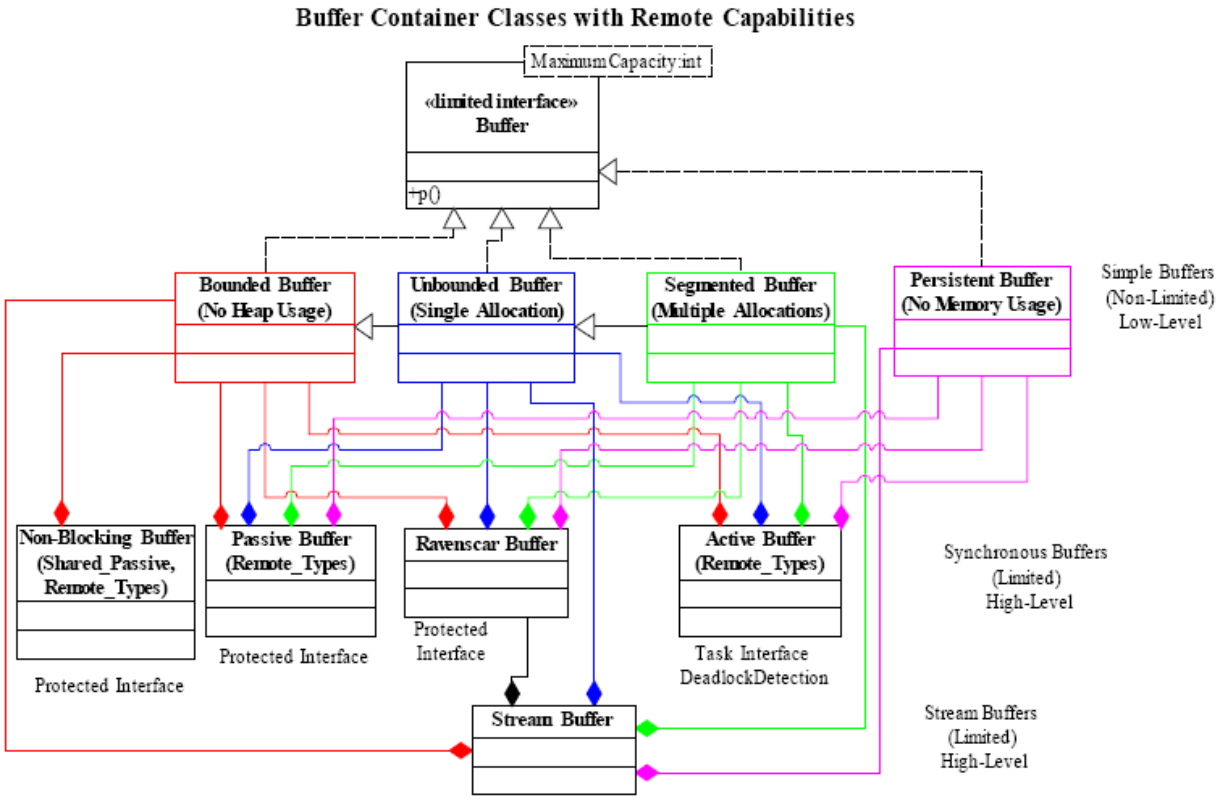

Figure 3 : Buffer Class Hierarchy

For example, if the driver of a vehicle locked his keys in the vehicle, he could call to a remote monitoring client through some means and request that the doors be unlocked remotely. This could result in the monitoring client sending an update to the vehicle to change the door lock status to the unlocked state. Once the vehicle received the update, it would report the new door lock status to interested clients.

Certain types of status could be disallowed from remote updates while others could be allowed. For example, it would not make sense for the amount of remaining fuel to be modified remotely.

4.1 Remote status buffer

For a vehicle to report a status update to a particular monitoring station, it needs to have a remote reference to the monitoring station. This is a challenge on a network where there are no central servers, and requires a discovery mechanism. This reference is a RACW value for the monitors message buffer which designates a limited interface to a Buffer.

The designated buffer object is a protected type implemented as a reusable container to ensure safe concurrent access. The vehicle application code is not linked to the specific buffer implementation, it only needs to be linked with the interface. This implies that different monitoring applications can use different buffer implementations so long as they are derived from the common Buffer interface.

The buffer itself needs to be able to store any status objects of a class hierarchy whereby a status object is the root class of the tagged type class hierarchy. Because various types of status objects can be stored in the buffer, the buffer needs to be able to store objects of varying sizes. This suggests the use of an Indefinite Buffer implementation that stores class-wide status objects. The interface for this buffer is instantiated in the following library level package.

```
with Status; use Status;
with Indefinite_Buffers;
package Status_Buffers is new
  Indefinite_Buffers
  (Element_Type => Status_Type'Class,
   Element_Index_Type => Positive);
pragma Pure (Status_Buffers);
```

4.2 Unicast vehicle access

The following package shows the remote unicast calls that can be made to a vehicle. This is a Remote_Types package that can be remotely accessed from the monitoring applications. Essentially, the Current_Status subprogram is used to request status records from the vehicle, and the Update subprogram is used to update a particular status record in the vehicle.

```
with Tag_Set;
   -- An Array of Tags used to select Status
   -- classes of interest
with Status; use Status;
with Status_Buffers; use Status_Buffers;
  -- Remote Client Buffer
private with Ada.Containers.Indefinite_Hashed_Maps;
  -- Ada 2005 features (private with and Containers)
package Vehicle_Server is
   pragma Preelaborate;
```

```
   pragma Remote_Types;
   -- Calls in this package can be made remotely
  subtype Vehicle_Id_Type is String (1 .. 20);
  type Client_Access is access all Status_Buffers.
       Remote_Buffer_Interface'Class;
  -- RACW to clients buffer
  type Vehicle_Type is tagged limited private;
  procedure Current_Status
   (Vehicle : Vehicle_Type;
    Client : Client_Access;
    Tags_Of_Interest : Tag_Set.Set);
   -- Request current snapshot of all status records
   -- in the vehicle that are inherited from specified tags
  procedure Update
   (Vehicle : in out Vehicle_Type;
    New_Status : Status_Type'Class);
   -- Update or add a status record in the vehicle
  function Vehicle_Serial_Number
   (Vehicle : Vehicle_Type)
     return Id_Type;
  procedure Is_Alive
   (Vehicle : access Vehicle_Type)
  is null;
  -- Null procedures are an Ada 2005 feature.
  -- Is_Alive is a null procedure but is still called remotely.
  -- If it can be called without raising an exception, then the
  -- communications are still good.
private
  use Ada.Containers;
  use type Status.Id_Type;
  -- The vehicles status objects are stored in an Ada 2005
  -- Hash table for lookups
  package Status_List is new
   Indefinite_Hashed_Maps
    (Key_Type => Id_Type,
     Element_Type => Status_Type'Class,
     Hash => Hash_Id,
     Equivalent_Keys => "="   );      --  "=" => <>
  -- Ensure safe concurrent access to the Status records
  protected type Status_List_Cache is
     procedure Update
      (Status_Item :
         in out Status_Type'Class);
     function Find (Status_Id : Id_Type)
       return Status_Type'Class;
  private
     Status_Items : Status_List.Map;
  end Status_List_Cache;
  type Vehicle_Type is tagged limited
   record
     Veh_Id : Vehicle_Id_Type;
     Serial_Number : Id_Type := 0;
     Current_Status_Set : Status_List_Cache;
  end record;
end Vehicle_Server;
```

4.3 Multicast vehicle access

Since this is a peer to peer network, there needs to be a way for a client to discover the vehicles on the network. It is also a better use of bandwidth if a status update going from a vehicle to interested clients is sent once via a multicast send, rather than separately to each client. Both these functions are achieved via two multicast addresses. One address is used for clients to send discovery requests to the set of vehicles on the network. The other address is used for vehicles to send status updates to the set of clients on the network.

Currently there are no DSA implementations that provide the support for multicast messaging, though the DSA interface does not seem to have any obstacles to prevent such an implementation. Multicast capabilities might be a good feature to add to enable better support for peer to peer programming. In the meantime, this can be worked around without too much effort while still using the DSA categorization pragmas to define the remote units as though multicast support were present.

General purpose messaging package

A general purpose package is defined to provide messaging capabilities. This provides an abstract type from which all multicast data types are derived. It also defines the endpoint type which defines the source and destinations of messages. An endpoint is a string that can either be an IP address with port number that designates a multicast address, or an endpoint may be of the form of a URI that defines a DTN endpoint.

```
package Messaging is
  pragma Pure;
  -- Endpoint may be in the form of a multicast IP address
  -- and port, or a DTN EID (Endpoint ID)  e.g.
  -- "239.255.128.128:55505"  or "dtn:\\foo.dtn\bar"
  subtype Endpoint_Type is String;
  -- Abstract type to be used to derive
  -- Multicast distributed data types
  type Messaging_Type
   is abstract tagged limited private;
  pragma Preelaborable_Initialization
   (Messaging_Type);
  procedure Subscribe
   (Subscriber : access Messaging_Type;
    Endpoint : Endpoint_Type) is null;
  -- Multicast Server side call to listen for incoming
  -- requests. This is null instead of abstract,
  -- because it is only implemented on the server side.
  -- This call typically just calls
  -- Messaging.Server.Subscribe providing a callback that
  -- invokes a local call on a multicast Remote_Types unit,
  -- by examining the Remote_Call_Data_Type record (see
  -- below) that was received on the multicast transport
  -- from the client.The callback knows how to interpret
  -- the specific derivation class of the
  --  Remote_Call_Data_Type interface.
  procedure Associate
   (Publisher : in out
      Messaging_Type'Class;
    Publisher_Endpoint : Endpoint_Type;
    Subscriber_Endpoint : Endpoint_Type);
  -- Multicast Client side call to associate a dummy
  -- multicast RACW object with a multicast endpoint
```

```
   type Remote_Call_Data_Type is
     interface;
   -- Call Parameter Data for remote calls. Not needed once
   -- DSA supports multicast. A specific multicast type
   -- defines what this object looks like.
   -- The corresponding server and client knows how to
   -- manipulate these objects
 private
   Max_Endpoint_Length : constant := 40;
    subtype Endpoint_Length is Positive
      range 1 .. Max_Endpoint_Length;
   type Messaging_Type
    is abstract tagged limited record
       Publisher,
       Subscriber : Endpoint_Type
        (1 .. Max_Endpoint_Length;
    end record;
  end Messaging;
```

The Subscribe call is a server side call that listens for incoming multicast requests. It is intended that the implementation of this call in the derived type call Messaging.Server.Subscribe with a callback that invokes a local call on a Remote_Types unit, by examining the Remote_Call_Data_Type record that was received on the multicast transport from the client side. The callback has knowledge on how to interpret the specific derivation class of the Remote_Call_Data_Type interface.

There is a client and a server component to this messaging capability, which are both implemented in corresponding child packages. The server (subscriber) listens for incoming messages on the associated multicast transport, while the client (publisher) sends multicast messages to the subscribers.

General purpose messaging client package

The Messaging Client package effectively issues a remote procedure call. Any parameter Call_Data passed to Publish is written to a communications transport using the call data structure provided.

```
package Messaging.Client is
   procedure Publish
     (Publisher : Messaging_Type'Class;
      Call_Data :  Remote_Call_Data_Type'Class);
   -- Make a remote multicast procedure call
end Messaging.Client;
```

General purpose messaging server

The Messaging Server package provides a callback that is used to take the parameter call data and generate the real remote call at the server site. The package is implemented as a task that monitors the multicast listener for incoming messages, and reads procedure call data from the stream.

```
package Messaging.Server is
   type Callback_Type is access procedure
     (Subscriber : Messaging_Type'Class;
      Parameters : Remote_Call_Data_Type'Class);
   -- Callback to implementation of actual multicast server
   -- object. Server Implementation uses Call Data
   -- parameter to generate the remote call
   procedure Subscribe
     (Subscriber :  access Messaging_Type'Class;
      Endpoint : Endpoint_Type;
      Callback : Callback_Type);
      -- The caller (A derived instance of
      -- Messaging_Type'Class that has invoked
      -- Messaging.Subscribe) specifies a callback that locally
      -- invokes a Remote call on a Remote Types unit, by
      -- interpreting the specific Remote_Call_Data_Type
      -- object received on a transport from a multicast client.
end Messaging.Server;
```

The caller that invokes Subscribe (A derived instance of Messaging_Type'Class) specifies a callback that locally invokes a Remote call on a Remote Types unit, by interpreting the specific Remote_Call_Data_Type object received on a socket from a multicast client.

Multicast vehicle discovery main package

Using the general purpose messaging packages, we now need to define the remote multicast access to the vehicle. This is still an abstract class, and is realized differently on both the client and server side with corresponding child packages.

```
with Messaging; use Messaging;
with Vehicle_Server; use Vehicle_Server;
with Status; use Status;
with Tag_Set;
package Vehicle_Discovery is
   pragma Remote_Types;
   type Vehicle_Discovery_Type
     is abstract limited new
        Messaging_Type with private;
   pragma Preelaborable_Initialization
     (Vehicle_Discovery_Type);
   not overriding  procedure Find
     (All_Vehicles : Vehicle_Discovery_Type;
      Client : Vehicle_Server.Client_Access;
      Serial_Number : Id_Type := 0;
      Tags_Of_Interest : Tag_Set.Set)
  is abstract;
   type Vehicle_Discovery_Access
     is access all
       Vehicle_Discovery_Type'Class;
   -- Pseudo RACW type providing access to
   -- all vehicles on the network
   pragma Asynchronous
     (Vehicle_Discovery_Access);
   -- Multicast calls are asynchronous in nature.
 private
   type Vehicle_Discovery_Type
     is abstract limited new
       Messaging_Type with null record;
   -- Structure used to contain data needed for remote calls
   type Vehicle_Call_Data
    is new Remote_Call_Data_Type with
     record
```

```
      Client : Vehicle_Server.Client_Access;
      Id : Id_Type;
      Tags_Of_Interest : Tag_Set.Set;
    end record;
end Vehicle_Discovery;
```

Vehicle discovery server package

There needs to be both a client and server side to the multicast discovery package. The Server side resides in a vehicle and responds to requests made from the monitoring station.

```
package Vehicle_Discovery.Responder is
  pragma Remote_Types;
  type Discovery_Responder_Type
   is limited new Vehicle_Discovery_Type
    with private;
  pragma Preelaborable_Initialization
    (Discovery_Responder_Type);
  overriding procedure Find
   (All_Vehicles : Discovery_Responder_Type;
    Client : Vehicle_Server.Client_Access;
    Serial_Number : Id_Type := 0;
    -- 0 means all vehicles, otherwise
    -- "discover" a specific vehicle.
    Tags_Of_Interest : Tag_Set.Set);
  -- Server side code sends all current status records to the
  -- specified regular RACW 'Client' using DSA
  -- unicast remote call.
  overriding procedure Subscribe
   (Subscriber :  access Discovery_Responder_Type;
    Endpoint : Endpoint_Type);
  -- Server side call to listen for incoming Find requests
private
  type Discovery_Responder_Type
   is limited new Vehicle_Discovery_Type
   with null record;
end Vehicle_Discovery.Responder;
```

Vehicle discovery client package

Similarly, on the client side, we need a corresponding package that issues a remote multicast call. Every remote call is converted to a Vehicle_Call_Data object and sent to the server by issuing the Messaging.Publish call. Note that the monitoring stations Client Buffer is passed as a parameter so that the Vehicle can respond directly to the requesting monitoring station.

```
package Vehicle_Discovery.Requestor is
  pragma Remote_Types;
  type Discovery_Requestor_Type
   is limited new Vehicle_Discovery_Type with private;

  pragma Preelaborable_Initialization
   (Discovery_Requestor_Type);
  overriding procedure Find
   (All_Vehicles : Discovery_Requestor_Type;
    Client : Vehicle_Server.Client_Access;
    Serial_Number : Id_Type := 0;
    Tags_Of_Interest : Tag_Set.Set);
  --  Send multicast request to retrieve current status for all
  --  vehicles on the network that have Tags_Of_Interest
  type Discovery_Requestor_Access
   is access all  Discovery_Requestor_Type'Class;
  -- Pseudo RACW type used to send
  -- requests to remote vehicles
   pragma Asynchronous
    (Discovery_Requestor_Access);
private
  type Discovery_Requestor_Type
   is limited new Vehicle_Discovery_Type
    with null record;
end Vehicle_Discovery.Requestor;
```

4.4 Status Record Hierarchy

Status objects are simply objects of a class hierarchy that can be extended to represent any information that needs to be stored in a vehicle, and sent to interested clients. The goal is to be able to pass around status objects that can be implemented in multiple object oriented programming languages such as Ada, C++, and Java. This way, the main applications can be written in a language of choice, but the objects are distributed as Ada objects by the DSA.

Clients and vehicles see and operate on the objects as though they were local objects. Updates to the objects are automatically distributed using infrastructure written in Ada.

This does present some challenges however. C++ and Java objects are not directly streamable in Ada using DSA. Ada does have excellent support for interfacing with other languages however [5]. Specifically, the GNAT compiler knows about the gcc C++ ABI (Application Binary Interface), which is also shared with the Gnu Java compiler. GNAT can import and export classes to these languages.

The technique employed is to derive Ada classes from C++ /Java classes, and pass them around using DSA. Code written in Java or C++ see these objects as Java/C++ objects. In this example, The Status classes are implemented in Ada and exported to C++, but it should also be possible to implement the status classes in C++ and import them into Ada.

Root status class in C++

First we show the Status base class written in C++. Status is an abstract class, so there is no need to derive an Ada class from the root Status class.

```
#ifndef STATUS_H
#define STATUS_H
#define INITIAL   0
#define CREATE    1
#define DELETE    2
#define UPDATE    3
#define DEPARTURE 4
class Status {
 public:
   static const char * className();
  // Get Ada external name of this class
   static void getListImage
```

```
        (Status & status,
         char *buf, int len);
      virtual int getStatusKind ();
      virtual void setStatusKind(int kind);
      virtual int reserved();
      int getId ();
      int getModificationCount ();
      virtual void setModificationCount (int count);
      int hash ();
      bool isEquivalent (Status & status);
    private:
      virtual void modify ();
      int kind;
      int identifier;
      int modified;
};
#endif
```

Vehicle status class in C++

Vehicle Status objects are status objects that reside in a vehicle and have a reportable status. Vehicle Status is also an abstract class, and similarly does not need an Ada derivation class to support streaming.

```
#ifndef VEHICLE_STATUS_H
#define VEHICLE_STATUS_H

#include "Status.h"

#define OPERATIONAL 0
#define FAULTY      1

class VehicleStatus : public Status {
  public:
    static char const * className();
    virtual int getState ();
    virtual void setState(int newState);

  private:
    int  state;
};
#endif
```

Door lock class in C++

A Door lock represents the status of a control device in the vehicle indicating whether the doors are locked or not.

```
#ifndef DOOR_LOCK_H
#define DOOR_LOCK_H
#include "VehicleStatus.h"
class DoorLock : public VehicleStatus {
  public:
    static char const * className();
  // Get the Ada external tag name for this class
    virtual bool isLocked ();
    virtual void lock();
    virtual void unlock();
  private:
    bool locked;
};
#endif
```

Ada version of root Status type

The corresponding stronger typed Ada version of this code, including the derived classes for distribution purposes is shown below. Note that non-virtual functions need to have a pragma export to setup the link name to match the mangled name that C++ is expecting to use. Virtual functions do not need this export, because they are accessed through virtual table structures in the object.

```
with Ada.Containers; use Ada.Containers;
package Status is
  pragma Pure;
  subtype Id_Type is Natural;
  subtype Update_Count_Type is Natural;

  type Status_Kinds is (Initial, Create, Delete, Update,
                        Departure);
  subtype List_Image_Type is String (1 .. 100);
  type Status_Type  is abstract tagged private;
  function Status_Kind (Status_Item : Status_Type)
    return Status_Kinds;
  pragma Export  (CPP, Status_Kind, "getStatusKind");
  procedure Set_Status_Kind
   (Status_Item : in out Status_Type;
    Kind : Status_Kinds);
  pragma Export  (CPP, Set_Status_Kind,  "setStatusKind");
  function List_Image (Status_Item : Status_Type)
     return List_Image_Type is abstract;
  function Id (Status_Item : Status_Type'Class)
     return Id_Type;
  pragma Export  (CPP, Id, "_ZN6Status5getIdEv");
  function Modification_Count
   (Status_Item : Status_Type'Class)
    return Update_Count_Type;
  pragma Export
  (CPP, Modification_Count,
   "_ZN6Status20getModificationCountEv");
  procedure Set_Modification_Count
   (Status_Item : in out Status_Type;
    Count : Update_Count_Type);
  pragma Export  (CPP, Set_Modification_Count,
                  "setModificationCount");
  function Hash_Id (Status_Id : Id_Type)
    return Hash_Type;
  function Hash  (Status_Item : Status_Type'Class)
    return Hash_Type;
  pragma Export (CPP, Hash, "hash");
  function Is_Equivalent  (Left, Right : Status_Type'Class)
    return Boolean;
  pragma Export (CPP, Is_Equivalent, "isEquivalent");

private
  procedure Modify  (Status_Item : in out Status_Type);
  pragma Export (CPP, Modify, "modify");

 type Status_Type is abstract tagged
   record
      Kind : Status_Kinds;
      Identifier : Id_Type;
```

```
    Modified : Update_Count_Type := 0;
  end record;
  pragma Convention (CPP, Status_Type);
end Status;
```

Ada version of vehicle status package

```
package Status.Vehicle is

   pragma Remote_Types;
   type Equipment_State_Type is
    (Operational, Faulty);
   type Vehicle_Equipment_Type
      is abstract new  Status_Type with private;
   function Operational_State
       (Vehicle_Equipment : Vehicle_Equipment_Type)
      return Equipment_State_Type;
   pragma Export (CPP, Operational_State, "getState");
   procedure Set_Operational_State
    (Vehicle_Equipment : in out Vehicle_Equipment_Type;
     New_State : Equipment_State_Type);
   pragma Export  (CPP, Set_Operational_State,
                   "setState");
  private
   type Vehicle_Equipment_Type is abstract
     new Status_Type with record
       State : Equipment_State_Type;
     end record;
   pragma Convention  (CPP, Vehicle_Equipment_Type);
  end Status.Vehicle;
```

Ada version of door lock status package

Since the Door Lock status is a concrete class, a true Ada class needs to be derived from the C++ class so that object of this class can be streamed using the DSA. The Ada version of the class, in this case Ada_Door_Lock_Type, does not add any additional components to the structure, but uses pragma Convention Ada to indicate to the compiler that this is a regular Ada object that should be streamable. Door_Lock_Type has the C++ (CPP) convention by default which it inherits from the Vehicle_Equipment_Type.

```
  package Status.Vehicle.Door_Locks is
    pragma Remote_Types;
    type Door_Lock_Type
      is new Vehicle_Equipment_Type  with private;
    function Is_Locked  (Door_Lock : Door_Lock_Type)
     return Boolean;
    procedure Lock_Doors
       (Door_Lock : in out Door_Lock_Type);
    procedure Unlock_Doors
      (Door_Lock : in out Door_Lock_Type);
    overriding function List_Image
       (Door_Lock : Door_Lock_Type)
            return List_Image_Type;
    type Ada_Door_Lock_Type
       is new Door_Lock_Type with private;

    -- Since this is a concrete type, we want to distribute such
    -- objects across the network. Therefore, we need to
    -- derive a true Ada class from the C++ type to
    -- enable C++ object to be distributed using DSA
    function Create
     (Status_Id : Id_Type;
      State : Equipment_State_Type;
      Locked : Boolean)
     return Ada_Door_Lock_Type;
    -- Objects are created as Ada objects, but will be viewed
    -- as C++/Java objects in those languages.
  private
    type Door_Lock_Type is new
     Vehicle_Equipment_Type with
      record
        Locked : Boolean;
      end record;
    -- The Ada Derived class should not need to add any
    -- components to the C++ class.
    type Ada_Door_Lock_Type is new
     Door_Lock_Type  with null record;
    pragma Convention (Ada, Ada_Door_Lock_Type);
  end Status.Vehicle.Door_Locks;
```

Provide external tag names to C++

A separate package is needed to return the static external tag names for C++, so that C++ clients can select the derivation classes of interest to be reported from the vehicles. This is a separate package because these declarations are not allowed in a Remote_Types package. Tag names are specified in C++ to allow monitoring stations to indicate which status types should be reported from a vehicle. Pragma Export statements are used to generate the C++ mangled name so that these subprograms can be accessed in C++.

```
  with Interfaces.C.Strings;
   use Interfaces.C.Strings;
  package Status.Class_Names is
    function Status_Class_Name  return chars_ptr;
    function Vehicle_Status_Class_Name  return chars_ptr;
    function GPS_Class_Name  return chars_ptr;
    function Fuel_Sensor_Class_Name  return chars_ptr;
    function Door_Lock_Class_Name  return chars_ptr;
    function Camera_Class_Name  return chars_ptr;
    function Summary_Class_Name  return chars_ptr;
    pragma Export (CPP, Status_Class_Name,
                   "_ZN6Status9classNameEv");
    pragma Export (CPP, Vehicle_Status_Class_Name,
                   "_ZN13VehicleStatus9classNameEv");
    pragma Export  (CPP, Door_Lock_Class_Name,
                    "_ZN8DoorLock9classNameEv");
    pragma Export (CPP, Summary_Class_Name,
                  "_ZN14VehicleSummary9classNameEv");
    pragma Export  (CPP, GPS_Class_Name,
                    "_ZN3GPS9classNameEv");
    pragma Export  (CPP, Camera_Class_Name,
                    "_ZN6Camera9classNameEv");
    pragma Export  (CPP, Fuel_Sensor_Class_Name,
                    "_ZN10FuelSensor9classNameEv");
  end Status.Class_Names;
```

5 Dispatcher (monitoring) design

A Dispatcher represents an application that monitors the status of all vehicles on the network. A dispatcher also has the capability of issuing commands to specific vehicles to provide controlling features.

When a Dispatcher joins the network it makes a multicast request to receive the current status from all vehicles currently on the network. Once this initial status has been received, the Dispatcher will continue to see updates for status records modified in any vehicles on the network. A dispatcher receives status updates through a local instance of a remote buffer object. The dispatcher application reads status update records received from the remote vehicles and processes the records accordingly.

5.1 Dispatcher client package

The Dispatcher_Client package is meant to provide the interface between the client application code and the distributed data replication system. In most cases, a single procedure call is provided to be called from multiple languages. In some cases, there are two versions of the same call when Ada language features can be used to provide a more convenient calling prototype for that language.

The design here is that client application code can register for change notification, by specifying tags of interest. If the object changed locally is derived from any of the tags in the Tags_Of_Interest, then the client notification callback is activated.

There may be multiple such registrations in a given client application at any given time. For example, there may be multiple windows displayed where each window displays a different set of data.

Dispatcher client specification

```
with Ada.Tags; use Ada.Tags;
with Interfaces.C.Strings;
use Interfaces.C, Interfaces.C.Strings;
with Status; use Status;
with Vehicle_Client;
with Status.Vehicle.Summary;
use Status.Vehicle,
   Status.Vehicle.Summary;
package Dispatcher is
   subtype Registration_Type is unsigned;
   subtype bool is unsigned_char;
   type Status_Callback_Type is not null
     access procedure
       (Status_Item : Status_Type'Class);
   -- Callback to client when a status changes
   procedure Query_All  -- For Ada callers
     (Tags_Of_Interest : Tag_Array;
      Process : Status_Callback_Type);
   -- Allows a client to process all locally cached objects
   -- using a client supplied access to subprogram
   -- parameter.
   procedure Query_All_C  -- For C callers
     (Tags_Of_Interest : chars_ptr_array;
      Process : Status_Callback_Type);
   function Exists      -- For Ada callers
     (Key : Id_Type)
       return Boolean;
   function Exists_C      -- For C Callers
     (Key : Id_Type)
       return bool;
   -- Indicates if a given status Id exists locally
   function Lookup (Key : Id_Type)
     return Vehicle_Client.Status_Type_Access;
   -- Returns the status item corresponding to the input
   -- Status Id Key.
   procedure Discover_Vehicles
     (Tags_Of_Interest : Tag_Array);
   -- Multicast request to discover vehicles on the network.
   -- Will result in vehicles responding with all status records
   -- that are derived from classes in the Tags_Of_Interest.
   procedure Register   -- For Ada Callers
     (Tags_Of_Interest : Tag_Array;
      Callback : Status_Callback_Type;
      Handle : out Registration_Type);
   -- Register client locally to receive change notifications
   -- for locally cached status objects that are derived from
   -- classes in the Tags_Of_Interest.
   procedure Register_C   -- For C Callers
     (Tags_Of_Interest : chars_ptr_array;
      Callback : Status_Callback_Type;
      Handle : out Registration_Type);
   procedure Request_Current_Status
     (Vehicle : Vehicle_Access;
      Tags_Of_Interest : Tag_Array);
   -- Remote call to a specific vehicle to retrieve current
   -- status that satisfy the Tags_Of_Interest
   procedure Deregister
     (Handle : Registration_Type);
   -- Cancel a local registration for change notification.
   function Update (Status_Item : Status_Type'Class)
      return Boolean;
   -- Update a status item remotely in a vehicle
   procedure Delete  (Status_Item : Status_Type'Class);
   -- Delete a status item remotely in a vehicle
private
   pragma Convention (C, Status_Callback_Type);
   pragma Export (C, Query_All_C, "queryAll");
   pragma Export (C, Exists_C, "exists");
   pragma Export (CPP, Lookup);
   pragma Export (C, Discover_Vehicles,
                  "discoverVehicles");
   pragma Export (C, Register_C, “registerClient");
   pragma Export (C, Deregister, "deregisterClient");
   pragma Export (CPP, Update, "updateItem");
   pragma Export (CPP, Delete, "deleteItem");
end Dispatcher;
```

Dispatcher client body

It is helpful to show small excerpts of the body of the Dispatcher client package to show how multicast dispatching is setup and how the remote buffer is declared.

```
with System.RPC;
-- Ada 2005 standard library package
with Ada.Environment_Variables; use Ada;
-- NOTE Buffer type, in this case it is a
-- Synchronous Unbounded Buffer
with Status_Buffers. Passive_Unbounded_Buffer;
use Status_Buffers.Passive_Unbounded_Buffer;
with Tag_Set;
with Messaging;
with Vehicle_Discovery.Requestor;
use Vehicle_Discovery.Requestor;
with Vehicle_Status.Subscriber;
-- Not shown but the Vehicle Status packages are similar
-- to the Vehicle_Discovery mechanism. Vehicles use this
-- for sending multicast status updates to dispatchers. The
-- server in this case is in the dispatcher, while the client
-- resides in the vehicle, in contrast to the vehicle
-- discovery packages, which has it the other way around.
package body Dispatcher is
  -- Declare the Status Buffer that is accessed remotely.
  -- This buffer is capable of storing 1000 Vehicle Status
  -- records, before blocking occurs.
  Buffer_Instance : aliased Buffer
   (Maximum_Capacity => 1_000);
  -- Create the RACW type for the buffer that can be
  -- distributed remotely to the vehicles for unicast calls.
  -- This is used when a vehicle responds to a multicast
  -- discovery request and sends status data back to the
  -- requesting dispatcher
  Client_Instance : Vehicle_Status.Status_Updater_Type
      (Buffer => Buffer_Instance'Access);
  -- Create a multicast server that
  -- receive status updates from vehicles
  Status_Subscriber : aliased
      Vehicle_Status.Subscriber. Subscriber_Type;
  -- Create a multicast client object that can be used to
  -- discover Vehicles on the network. This is declared as a
  -- RACW, and represents all vehicles on the network. It is
  -- not really a RACW, we use the multicast packages to
  -- send the data, but conceptually it can be thought
  -- of a special kind of RACW
  Vehicle_Discovery : aliased Discovery_Requestor_Type;
  All_Vehicles : constant Discovery_Requestor_Access
   := Vehicle_Discovery'Access;
  Initialized : Boolean;
-- ...
  procedure Discover_Vehicles
   (Tags_Of_Interest : Tag_Array) is
  begin
    if not Initialized then
      --  Setup Publisher for vehicle discovery requests
      if Environment_Variables.Exists
       ("USE_DTN") then
        -- Use a DTN transport
       Messaging.Associate
        (Publisher => Vehicle_Discovery,
          Publisher_Endpoint => "dtn://status-monitors.dtn/?"
         Subscriber_Endpoint => "dtn://vehicles.dtn/*");

        Vehicle_Status.Subscriber. Subscribe
          (Subscriber => Status_Subscriber'Access,
           Endpoint => "dtn://status-monitors.dtn/*");
       else
        -- Use simple multicast transport
        -- more suitable for wired network
         Messaging.Associate
          (Publisher => Vehicle_Discovery,
           Subscriber_Endpoint => "239.255.128.129:55507");
          Vehicle_Status.Subscriber. Subscribe
           (Subscriber => Status_Subscriber'Access,
            Endpoint => "239.255.128.128:55505");
         end if;
       Initialized := True;
      end if;
      --  Issue Multicast request to  Discover Vehicles
      All_Vehicles.Find
       (Client => Buffer_Instance'Access,
        Tags_Of_Interest  =>  Create_Tag_Set
          (Tag_Array'(1 => Vehicle_Equipment_Type'Tag)));
    end Discover_Vehicles;
    -- Unicast request for status from a remote vehicle. Note
    -- we pass the clients Buffer as a RACW so that the
    -- vehicle can respond directly to this dispatcher.
    procedure Request_Current_Status
     (Vehicle : Vehicle_Access;
      Tags_Of_Interest : Tag_Array) is
    begin
      -- Send RACW buffer to remote vehicle
      Vehicle.Current_Status
       (Client => Buffer_Instance'Access,
        Tags_Of_Interest =>  Create_Tag_Set
          (Tags_Of_Interest));
    exception
      when
        System.RPC.Communication_Error =>  Put_Line
         ("Vehicle died, " &
          "Remote Call Failed");
    end Request_Current_Status;
  end Dispatcher;
```

5.2 Remote buffer implementation

As mentioned previously, Vehicles communicate with clients by writing status objects into the remote clients buffer, which is passed to the Vehicle as a Remote Access to Class-Wide type (RACW) object.

Shown earlier was the instantiation of the Status Buffer interface for the Indefinite Buffer of Class-Wide Status records. On the client side, we now discuss the implementation of the remote Buffering. Because the Buffer is accessed remotely, this suggests that the buffer implementation needs to be one of the synchronous buffer classes to ensure safe concurrency. Since the number of vehicles on the network is unknown, and because the dispatcher is likely running on a desktop computer where memory use is not restricted, the Unbounded Buffer implementation might be a better choice than a Bounded Buffer. A good choice in this case would be to use a Passive Unbounded Buffer, which will only use as much memory as needed to handle peak demands, and can be

resized to smaller allocation during periods of low activity. A Passive buffer supports concurrency which is needed since there can be multiple writers and a reader accessing the buffer at the same time.

Passive Unbounded Status Buffer Package

The Passive Unbounded Status Buffer is a blocking Unbounded Buffer that supports multiple readers and writers.

```
with Indefinite_Buffers.Passive_Unbounded;
package Status_Buffers. Passive_Unbounded_Buffer
  is new  Status_Buffers.Passive_Unbounded;
pragma Preelaborate
  (Status_Buffers.Passive_Unbounded_Buffer);
pragma Remote_Types
  (Status_Buffers.Passive_Unbounded_Buffer);
```

6 DTN alternative to multicast

The reusable messaging transport described previously shows support for two types of transport. A simple multicast transport is implemented, and a DTN transport. The simple multicast transport is not reliable however, and is intended to work in a wired network scenario where all vehicles and monitors can see each other with good connectivity. The DTN transport is better suited for a real mobile environment where good connectivity between nodes cannot be assumed, and end to end connectivity between all nodes is also not likely, given the nature of radio communications spread over a larger area, where terrain and obstacles can provide barriers to communication. Communications in heterogeneous networking environments can break down [6] particularly when end to end connectivity between nodes is not guaranteed or too sporadic for reliable use.

Technologies that allow communications in Delay Tolerant Network (DTN) environments are a subject of active research. The Delay Tolerant Networking Research Group (http://www.dtnrg.org/wiki) represents a community of researchers and industry dedicated to this topic. Their research centers around RFC 4838 and the experimental bundle protocol. There exists several publicly available implementations of this protocol. Such technologies are being considered for a wide variety of use including deep space communications, tactical military networks, sensor networks, and underwater acoustic networks.

These environments suggest a good fit for the Ada programming language, in particular situations where high reliability is needed. Though this may be a good niche area for Ada, the author is not aware of any Ada implementations available for use. The main thrust of DTN development appears to consist of DTN2, written in C++, and ION, written in C. In addition, there exists a couple of Java implementations of the bundle protocol, and another one written in Python. If the use of Ada is to be encouraged in tactical networking and spacecrafts, then there needs to be a DTN implementation made available for general use. A good first step would be to provide Ada bindings to the DTN2 reference implementation. Such bindings were produced for the prototype associated with this paper, and it is hoped that these bindings can be made publicly available in 2009.

7 Conclusions

Ada 2005 has language features that facilitate solutions for the distributed programming problem domain with an economy and ease of expression. The Distributed Systems Annex can be used to distribute objects to applications written in other languages including C++ and Java if the compilers for the respective languages share a common ABI. There is a window of opportunity to provide an Ada implementation of the bundle protocol to support communications in challenged DTN networking environments. This would be a natural fit for extending Ada's niche area of use to include reliable communications to spacecraft and tactical military networking environments.

References

[1] Warthman, F., Delay-Tolerant Networks (DTNs): A Tutorial v1.1, Mar 2003 (http://www.dtnrg.org/docs/tutorials/warthman-1.1.pdf

[2] Moore, B.J., A Buffer Container Class Hierarchy using Ada 2005, SIGAda'08, Proceedings of the 2008 ACM SIGAda Annual International Conference

[3] Miranda, J., Schonberg, E. (March 21, 2007) Abstract Interface Types in GNAT: Conversions, Discriminants, and C++. Adacore Technical Paper "http://www.adacore.com/wp-content/uploads/2007/03/ifaces_ae06.pdf"

[4] Comar, C., Gingell, M., Hainque, O., Miranda, J. (July 20, 2006) Multi-Language Programming: The Challenge and Promise of Class-Level Interfacing. Adacore Technical Paper, "http://www.adacore.com/wp-content/uploads/2006/07/Class_level_interfacing.pdf"

[5] Taft, S.T., Duff, R. A., Bruckardt, R.L. And Plödereder, E. Eds (2000). Consolidated Ada Reference Manual. LNCS 2219, Springer-Verlag

[6] V. Serf et al, Delay-Tolerant Networking Architecture, RFC 4838, "http://www.ietf.org/rfc/rfc4838.txt"

Ada Gems

The following contributions are taken from the AdaCore Gem of the Week series. The full collection of gems, discussion and related files, can be found at http://www.adacore.com/category/developers-center/gems/.

Gem #6: The Ada95 Multiple Views Idiom vs. Ada05 Interfaces

Matthew Heaney, On2 Technologies

Date: 18 June 2007

Abstract: The multiple views idiom is a technique that allows you to create the effect of deriving from multiple tagged types simultaneously. It is a powerful mechanism for composing abstractions that complements interface types.

Let's get started…

One of the major changes to the Ada language is the addition of interfaces, stateless types that specify a set of operations. Like a tagged type, you can derive from an interface type, but unlike a tagged type, you can derive from multiple interfaces simultaneously.

Typically you use an interface type as a kind of specification. An abstraction can be written in terms of an interface type, which makes the abstraction completely general, in that it can be used with any type that implements that interface.

Ada95 does not have interface types, so an obvious question is, Can you create the effect of deriving from multiple interface types, but in Ada95? The answer is yes, using a technique called the "multiple views" idiom. The technique makes it possible for a type to provide different views of itself, with each view having a different type. This is very much like having multiple interfaces, but you have to build the infrastructure yourself.

To illustrate the difference between interfaces and multiple views, we'll first design a simple abstraction for persistence in terms of an interface type, and then redesign it using the multiple views technique. An interface for persistence would look something like this:

```
package Persistence_Types2 is
  type Persistence_Type is limited interface;

  procedure Write
    (Persistence : in Persistence_Type;
     Stream      : not null access Root_Stream_Type'Class)
     is abstract;
  … -- Read not shown here
end Persistence_Types2;
```

Any type that derives from Persistence_Type is saying that it can be saved out to (and loaded in from) some persistent medium. Deriving from the interface type is easy:

```
package P2 is
  type T is limited new Persistence_Type
     with null record;

  overriding
  procedure Write
    (Persistence : in T;
     Stream      : not null access
                   Root_Stream_Type'Class);
  …
end P2;
```

Now all the application has to do is provide a persistence mechanism, that accepts an object that supports the persistence interface. Here's our abstraction for doing that:

```
package Persistence_IO2 is
  … -- Initialization details omitted here
    procedure Save (Persistence : in
                    Persistence_Type'Class);
end Persistence_IO2;
```

This provides the infrastructure that allows any type (here, type T) that derives from Persistence_Type to be written to the persistence medium. A file-based implementation might look something like this:

```
package body Persistence_IO2 is
  File : Ada.Streams.Stream_IO.File_Type;
  …
  procedure Save
    (Persistence : in Persistence_Type'Class)  is
  begin
    Persistence.Write (Stream (File));
  end Save;
end Persistence_IO2;
```

Now we did all this using an Ada05 interface type, but nothing we did strictly requires an interface type. You can achieve the same effect using a tagged type, which is how it would have been done in Ada95. The "interface" is just an abstract tagged null record:

```
package Persistence_Types is
  type Persistence_Type is
    abstract tagged limited null record;
  procedure Write
    (Persistence : in Persistence_Type;
     Stream      : access Root_Stream_Type'Class)
      is abstract;
  … -- Read omitted here
end Persistence_Types;
```

Here the Persistence_Type is a tagged type instead of an interface type, but it's otherwise the same. Even the Persistence_IO abstraction is the same as before. What is different is how the type is used to support persistence in some other type. That other type will provide a "persistence view" of itself. We wish for the type to support multiple views (just as it would were it implementing multiple interfaces), so it's not a simple matter of directly deriving from persistence type,

since Ada does not support derivation from more than one tagged type (this is true of both Ada95 and Ada05).

What we will do to implement a persistence view is to create an intermediary type that derives from Persistence_Type, but with an access discriminant that designates the parent (the type that supports the view). This allows operations of the intermediary type to bind to the parent type, since the discriminant provides access to the parent's state. To see how this all works, let us first show what the public part of the parent type looks like:

```
package P is
  type T is limited private;

  type Persistence_Class_Access is
    access all Persistence_Type'Class;

  function Persistence (Object : access T)
    return Persistence_Class_Access;
  … others views would be declared here
private
  … -- see text below
end P;
```

Type T supports persistence by providing a Persistence function that returns an access value designating an instance of Persistence_Type. The "persistence view" of type T is obtained by invoking this accessor function. This mechanism can be extended any number of times, by providing multiple accessor functions that each return distinct views.

The Persistence function returns an access value that actually designates a component of record T. The type of the component is the intermediary type we mentioned earlier, that derives from Persistence_Type, declared like this:

```
private
  type Persistence_View (Object : access T) is
    new Persistence_Type with null record;  -- no state
                                            -- req'd here
  procedure Write
    (Persistence : in Persistence_View;
     Stream      : access Root_Stream_Type'Class);
```

Note that the Persistence_View type is a null extension (it doesn't require any state of its own), but with an access discriminant that designates the parent type T. This allows the implementation of operation Write to see the representation of the parent type, since the Persistence parameter has an access discriminant that designates the T instance, and so the operation has access to all of the state that needs to be written to persistent storage.

The parent type T is implemented by declaring an aliased component of the intermediary type:

```
  type T is limited record
     Persistence : aliased Persistence_View (T'Access);
    … -- rest of state here
  end record;
```

The Persistence component is aliased since function Persistence returns an access value designating that component:

```
  function Persistence (Object : access T)
    return Persistence_Class_Access  is
  begin
     return Object.Persistence'Access;
  end;
```

The other difference between interface types and the multiple views idiom is how a client would actually invoke the operation to perform the persistence operation. In the interface case it's simple: type T derives from Persistence_Type directly, so instances of type T can be passed to any operation whose type is Persistence_Type'Class. The Persistence_IO package has just such an operation, so the call would look like this:

```
procedure Test_Persistence2 is
  Object : T;
begin
  …
  Persistence_IO2.Save (Object);
end;
```

You have to do a little more work in the multiple views case, since the conversion from T to Persistence_Type isn't automatic. Here we need to explicitly invoke the Persistence function to "convert" from type T to Persistence_Type'Class. The accessor function has an access parameter and so we must declare the instance of type T as aliased. The call looks like this:

```
procedure Test_Persistence is
  Object : aliased T;
begin
  …
  Persistence_IO.Save (Persistence (Object'Access).all);
end;
```

That's all there is to it. The multiple views idiom is actually a very powerful mechanism for composing abstractions. I have characterized the technique as an Ada95 idiom, but note that even in Ada05 it is occasionally useful, such as when you need to mix a tagged type into an existing hierarchy. The most common example is needing to add controlledness to a leaf type in a class, because the root type doesn't itself derive from controlled.

Gem #48: Extending Interfaces in Ada 2005

Quentin Ochem, AdaCore

Date: 13 October 2008

Abstract: Ada 2005 introduced the notion of interfaces for designing object classes. While interfaces are extremely convenient for implementing new hierarchies, they can be difficult to extend once they've started to be used. The addition of a new primitive can break all type derivations, as a type has to implement all abstract primitives inherited from its parents. In this Gem, we'll see two ways to overcome this problem, one OOP-generic, and one specific to Ada 2005.

Let's get started…

The classic OOP way

Let's assume we have the following interface:

```
type Animal is interface;
procedure Eat (Beast : in out Animal) is abstract;
```

All types implementing the Animal interface have to override the Eat operation:

```
type Cat is new Animal with record …
procedure Eat (Beast : in out Cat);
```

Now, after a while, the developer of Animal might feel the need to let animals eat something specific, and would like to add the following operation to the interface:

```
procedure Eat (Beast : in out Animal;
               Thing : in out A_Thing);
```

Unfortunately, there are hundreds of species of animals implementing this interface, and having to migrate everything will be too painful. Not to mention that most of them don't even need this new way of eating - they're just happy eating some random amount of anonymous food. Extending this interface is just not the way to go - so the extension has to be done separately, in a new interface, such as:

```
type Animal_Extension_1 is interface;
procedure Eat (Beast : in out Animal_Extension_1;
               Thing : in out A_Thing) is abstract;
```

So now, Animals that need to rely on this new way of eating will need to be declared, such as:

```
type Cat is new Animal and Animal_Extension_1
   with record …
```

Note that it's even possible to enforce the fact that an extension of Animal has to be an Animal in the first place, by writing:

```
type Animal_Extension_1 is interface and Animal;
```

which will lead to a simpler declaration for type Cat, as there's no longer a need to extend from two interfaces:

```
type Cat is new Animal_Extension_1 with record …
```

The rest of the code will remain completely untouched thanks to this change. Calls to the new subprogram will require some additional amount of work though, as we'll first have to check that the type of an Animal that we're dealing with is indeed a descendant of Animal_Extension_1, and perform a conversion to that interface's class, before calling the new version of Eat:

```
The_Animal : Animal'Class := ...
if The_Animal in Animal_Extension_1'Class then
  Animal_Extension_1'Class (The_Animal).
    Eat (Something);
end if;
```

The Ada 2005 Way

Ada 2005 introduces the notion of null procedures. A null procedure is a procedure that is declared using "is null" and logically has an empty body. Fortunately, null procedures are allowed in interface definitions - they define the default behavior of such a subprogram as doing nothing. Back to the Animal example, the programmer can declare the interface's Eat primitive as follows:

```
procedure Eat (Beast : in out Animal;
               Thing : in out A_Thing) is null;
```

All of our hundreds of kinds of animals will automatically inherit from this procedure, but won't have to implement it. The addition of this declaration does not break source compatibility with the contract of the Animal interface. Moreover, as no new types are involved, it's a lot easier to make calls to this subprogram - no more need to check membership or write a type conversion, and we can just write:

```
The_Animal : Animal'Class := ...
The_Animal.Eat (Something);
```

which will execute as a no-op except for animals that have explicitly overridden the primitive.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o K.U. Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Peter Dencker
Steinäckerstr. 25
D-76275 Ettlingen-Spessartt
Germany
Email: dencker@web.de
URL: ada-deutschland.de

Ada-France

Association Ada-France
c/o Jérôme Hugues
Département Informatique et Réseau
École Nationale Supérieure des Télécomunications
46, rue Barrault
75634 Paris Cedex 135
France
Email: bureau@ada-france.org
URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
Ada-Spain
P.O.Box 50.403
28080-Madrid
Spain
Phone: +34-942-201-394
Fax: +34-942-201-402
Email: gutierjj@unican.es
URL: www.adaspain.org

Ada in Sweden

attn. Rei Stråhle
Saab Systems
S:t Olofsgatan 9A
SE-753 21 Uppsala
Sweden
Phone: +46 73 437 7124
Fax: +46 85 808 7260
Email: Rei.Strahle@saabgroup.com
URL: www.ada-i-sverige.se

Ada Switzerland

attn. Ahlan Marriott
White Elephant GmbH
Postfach 327
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: ada@white-elephant.ch
URL: www.ada-switzerland.ch