ADA USER JOURNAL

Volume 31
Number 4
December 2010

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.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

The majority of the content of this last issue of 2010 is the result of the work of two groups of Ada practitioners which devote part of their time to the development and promotion of the Ada language within two of the topics where Ada is without doubts a strong technology: real-time systems and reliable software.

In the first part of the issue, we present an overview of the 14th International Real-Time Ada Workshop (IRTAW), which was held approximately one year ago, in Portovenere, Italy. This overview provides not only the workshop's session summaries, allowing readers to get the necessary insight in the fruitful discussions and results of the workshop, but also a post workshop report, by Alan Burns, of the University of York, UK, presenting how the workshop proposals are going through the Ada 2012 standardization process (since many times we wonder what happens to the Ada Issues that the workshop produces). IRTAW has been and still is a major forum for Ada evolution (and not only in the real-time arena); it is therefore with happiness that in the issue we also have the announcement of the next edition of the workshop, which will take place in September 2011, in the mountains nearby Santander, Spain.

In the second part o the issue, we conclude the publication of the results of the work of the Software Vulnerabilities group, which produced the Ada annex (published in the September issue of the Ada User Journal) to the ISO Technical Report on "Avoiding Programming Language Vulnerabilities through Language Selection and Use" (ISO/IEC PDTR 24772.2). To finalize the work, in this issue we publish the annex concerning SPARK.

I would also like to point out to our readers the forthcoming events section, where, before the IRTAW call for papers, we already have preliminary information concerning "The Ada Connection", an event that combines the Ada-Europe 2011 conference and the Ada-Conference UK 2011, taking place in Edinburgh, Scotland, next June 20-24.

In the Ada Gems section, the issue publishes the series of gems concerning the Distributed Systems Annex, by Thomas Quinot, of AdaCore, France. Last, but definitely not least, readers will find the wealthy of information in the usual news digest and calendar sections.

In a final note, this last 2010 issue of the Ada User Journal should be reaching you already in 2011. Nevertheless, hopefully there was no substantial delay (except for the normal printing and distribution time). We are making our best efforts in putting the AUJ back in its scheduled publication date, and I would like to thank the Editorial Team of the Journal – Jorge Real (Deputy Editor), Dirk Craeynest (Calendar and Events), and Marco Panunzio (News Digest) – for all their volunteer efforts in achieving this.

Our best wishes for 2011,

Luís Miguel Pinho
Porto
December 2010
Email: lmp@isep.ipp.pt

Quarterly News Digest

Marco Panunzio

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

Contents

Ada-related Organizations

Video and PDF of presentations at the Ada-Europe 2010 conference

From: Thomas Løcke <tl@ada-dk.org>
Date: Tue, 19 Oct 2010
Subject: Ada-Europe 2010 - the videos
URL: http://ada-dk.org/?page=news&news_id=194

Yesterday Jorge Real (Ada-Europe General Secretary) informed me that the Ada-Europe 2010 conference talks are now online as PDF's and/or video.

To that I only have one thing to say: Great!

No less than 33 talks are available for download, so if you're interested in Ada, then the odds are good for finding something of interest to you.

Now if you'll excuse me, I'll go pour myself a cup of tea, grab a comfortable chair and start watching some Ada talks. If all goes well, I'll come out of it a bit wiser.

A big thank you to Ada-Europe for making all this material available, and to the authors for permitting it.

[you can find the material at http://www.disca.upv.es/jorge/ae2010/outcome.html —mp]

New website for Ada-France

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 03 Nov 2010 16:16:58 +0100
Subject: New site for Ada-France
Newsgroups: comp.lang.ada

Ada-France is pleased to announce the total revamping of its website, with up-to-date information, new look-and-fill, twitter-indeti.ca-facebook-RSS feeds, and more!

If you speak (or at least understand ;-)) French, please visit us at:

http://www.ada-france.fr/

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Wed, 03 Nov 2010 16:41:54 +0100
Subject: Re: New site for Ada-France
Newsgroups: comp.lang.ada

[…]

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

I guess? Otherwise, this would have been a literally cosmetic change. :-)

Les voilà les accents! Good to see the correct characters.

From: J-P. Rosen <rosen@adalog.fr>
Date: Thu, 04 Nov 2010 11:50:24 +0100
Subject: Re: New site for Ada-France
Newsgroups: comp.lang.ada

[…]

Ooops, yes of course, sorry for the confusion.

First open meeting of "Ada in Denmark"

From: Thomas Løcke <tl@ada-dk.org>
Date: Thu, 04 Nov 2010
Subject: An open Ada-DK meeting!
URL: http://ada-dk.org/?page=news&news_id=204

December 7th, 2010 marks the day when the first open Ada-DK meeting is being held.

The "open" part means that the meeting is not a member only affair, but that anybody interested in Ada is welcome, so feel free to invite whomever you might believe could be interested in spending an evening talking about Ada.

We do not yet have an agenda for the meeting, simply because we don't know how many are going to attend. It might be a smashing hit, a massive failure or something in between. Time will tell. If it's not a complete failure, I'm certain we'll come up with an agenda for the next meeting.

The plan is to have one open Ada-DK meeting each month, on the first Tuesday of the month.

If you're interested in participating, feel free to send us an email, and we'll inform you of the when and where. Or join the Freenode IRC #ada channel and look for ThomasLocke.

The meeting is of course free.

[contact Ada-DK at info@ada-dk.org —mp]

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]

Workshop on Object-Oriented technologies and Ada in High-Integrity Systems

From: J-P. Rosen <rosen@adalog.fr>
Date: Tue, 07 Sep 2010 16:54:41 +0200
Subject: Ann: Workshop on OO and Ada in High Integrity Systems
Newsgroups: comp.lang.ada

There will be a workshop during SIGAda, whose goal is to define a set of Ada restrictions that would make using Object Oriented technologies with Ada more applicable to high integrity systems. The workshop could lead to the definition of a profile and to a document explaining and justifying the profile, following the example of the Ravenscar profile for the concurrency aspects.

All those interested are invited:

- to attend

- and/or send suggestions for possible restrictions (with justification)

to rosen@adalog.fr

The workshop will start with a summary of received proposals, followed by discussion and (hopefully!) agreement on the content of the profile.

Those who are interested in the topic but are not aware of all the issues are invited to attend the tutorial on the same topic during the conference.

More info on http://www.sigada.org/conf/sigada2010/

Please pass the word around, and see you in Fairfax!

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 08 Sep 2010 07:28:01 +0200

Subject: Re: Ann: Workshop on OO and Ada in High Integrity Systems
Newsgroups: comp.lang.ada

> […] Hope there will be some on-line publications after this meeting (just a wish).

> […]

This is definitely the intent.

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 15 Oct 2010 12:25:35 +0200
Subject: Material for the HR-OO workshop at SIGAda
Newsgroups: comp.lang.ada

I have made a page in preparation for the High-Reliability Object-Oriented Ada workshop at SIGAda:

http://www.adalog.fr/hr-oo-workshop/

If you plan to attend, there is no requirement to send a formal position paper, but it would be nice if you could send your views in advance as a proposal, as explained in the above page.

If you don't plan to attend, you are also welcome to send proposals and good ideas!

See you in Fairfax.

Ada-Europe Programming Contest "The Ada Way"

From: Dirk Craeynest <dirk@cs.kuleuven.ac.be>
Date: Mon, 27 Sep 2010 20:22:27 +0000 UTC
Subject: Press Release - Ada-Europe Kicks Off First Programming Contest
Newsgroups: comp.lang.ada, fr.comp.lang.ada,comp.lang.misc

FOR IMMEDIATE RELEASE

Ada-Europe Kicks Off its First Annual Student Programming Contest "The Ada Way"

Brussels, Belgium (September 28, 2010) - Ada-Europe[1], the international organization that promotes the knowledge and use of the Ada programming language in European academia, research and industry, is pleased to announce "The Ada Way"[2]. This annual student programming contest aims to attract students and educators to Ada in a form that is both fun and instructive. Entries are now open for the 2010-11 competition and judging takes place in May next year.

In line with the start of the football season, this year's challenge is to build a software simulator of a football match. The software system, programmed in Ada, will need to support a number of gaming and football features including speed, tactical skills and player fatigue.

The submitted code will include a software core, implementing the logic of the simulation, as well as read-write graphical panels for participating football team managers.

Candidate submissions will be judged on a number of evaluation criteria including:

- Coverage of requirements.
- Syntactic, semantic, programmatic and design correctness.
- Clarity and readability of the code.
- Quality of design.
- Ingenuity and cuteness of the solution.
- Time and space efficiency of the solution.

The winning submission will win a framed award, one free registration and up to 3 reduced student fees for representatives of the winning team to attend to the Ada-Europe 2011 Conference[3], accommodation and airfare for the team representatives, an exhibition slot in the conference program, and visibility in electronic and printed media.

This year's competition is sponsored by Ada-Europe, AdaCore, and Atego.

To enter, and for the full specification and details of software requirements, please go to the official web site of "The Ada Way", www.ada-europe.org/AdaWay.

About Ada-Europe

Ada-Europe is the international non-profit organization that promotes the knowledge and use of the Ada programming language in academia, research and industry in Europe. Ada-Europe has member organizations all over the continent, in Belgium, Denmark, France, Germany, Spain, Sweden, Switzerland, as well as individual members in many other countries.

A PDF version of this press release is available at www.ada-europe.org.

Press contact

Dirk Craeynest, Ada-Europe Vice-President,
Dirk.Craeynest@cs.kuleuven.be

[1] www.ada-europe.org

[2] www.ada-europe.org/AdaWay

[3] www.ada-europe.org/conference2011

Ada at FOSDEM 2011

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 6 Oct 2010 02:28:49 -0700 PDT
Subject: Call for speakers for Ada at FOSDEM 2011; deadline on 2010-10-16!
Newsgroups: comp.lang.ada

The next FOSDEM[1], the Free and Open-Source European Developers'Meeting, will take place on Saturday 5 and Sunday 6 February 2011 in Brussels, Belgium. The event is entirely free; speakers and attendees alike are unpaid but enthusiastic volunteers. The audience consists mostly of fellow developers. FOSDEM is one of the major events giving visibility to worthy projects and languages. Ada-Belgium organized two very successful series of presentations at FOSDEM 2006[2] and FOSDEM 2009[3], earning a good reputation for reliability and quality, and is now seeking speakers for FOSDEM 2011. Two speakers have already volunteered.

[1] http://www.fosdem.org

[2] http://archive.fosdem.org/2006/2006/index/dev_room_ada.html

[3] http://people.cs.kuleuven.be/~dirk.craeynest/ada-belgium/events/09/090207-fosdem.html

For several years now, Ada-Belgium has been running a dedicated mailing list for the coordination of Ada at FOSDEM [4]. If you would like to be a speaker or just attend the event, please subscribe to this mailing list and peruse the archives; traffic is low and the signal-to-noise ratio is extremely high :) We can discuss accommodation and catering issues as well as technical ones on this list. A proposal should consist of a 10-line summary of the presentation, a 10-line biography of the speaker and an optional small picture (in JPEG or PNG) of the speaker.

The deadline for the submission of proposals for a Developers' Room at FOSDEM is ten (10) days from now on October 16, 2010 [5]. If you would like to be a speaker, please react now on the adafosdem list so we can arrange for a joint proposal for all Adaists.

[4] http://listserv.cc.kuleuven.be/archives/adafosdem.html

[5] http://www.fosdem.org/2011/call_for_mainspeakers_devrooms

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Date: Wed, 27 Oct 2010 23:44 CEST
Subject: Again NO Ada at FOSDEM (was: Call for speakers for Ada at FOSDEM 2011)
Mailing list: ada-belgium-info@cs.kuleuven.be

Dear Ada-Belgium friend,

This is a short status report on what has happened since we posted the appended Call for Speakers earlier this month.

Based on the feedback we received, we prepared a detailed proposal for an Ada Developer Room during both days of FOSDEM 2011 next February.

Our proposal, including the full list of presentations and speakers, is available on the Ada at FOSDEM 2011 web-page at

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/11/110205-fosdem.html

Unfortunately, earlier today Valentine wrote to the AdaFOSDEM mailing list:

"Hi all

bad news …
they did not send a mail to say our request was rejected…
but as far as i can read …

:/

http://www.fosdem.org/2011/news/accepted-devrooms"

The list of accepted DevRooms does not include Ada, so after very successful and well attended Ada DevRooms in 2006 and 2009, we once more didn't make it for 2011.

Thanks again, everybody involved with the proposal this year.

Maybe we should reconsider and start looking for another event than FOSDEM for next year…

Valentine, Ludovic, Dirk

The FOSDEM Team of Ada-Belgium

Ada Semantic Interface Specification (ASIS)

On inconsistencies in ASIS type definitions and ASIS versions

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Sun, 07 Nov 2010 09:04:11 +0100
Subject: Asis.Text.Character_Position : inconsistency ?
Newsgroups: comp.lang.ada

Hello Ada writers,

I was trying to patch AdaControl (http://www.adalog.fr/adacontrol2.htm) so that it can build with Gela-ASIS. This is an interesting process of little refactoring, as Gela-ASIS is clearly designed so that it make no reference at all to a specific compiler implementation (it do so defining an Implementation_Defined_Type from which other discrete types are derived). After a first attempt I restarted again and felt something was wrong as I could not get something coherent about types.

Prior note: I do not own a copy of the ISO ASIS reference, so I cannot tell who of TenDRA or ISO is right or wrong.

What disturbs me, if I believe (as I feel it is) that Gela-ASIS heavily stick on the ASIS standard: Asis.Text defines two types, Character_Position and Line_Number. Character_Position is derived from ASIS_Natural, which in turn is derived from ASIS_Integer, which, the latter, is implementation defined.

Also, ASIS defines a type Asis.Program_Text, which is not implementation defined, and which is explicitly an unconstrained subtype of the Ada's type Standard.Wide_String.

Shouldn't be Asis.Text.Character_Position be defined so that is can be used as an index in Asis.Program_Text? This would imply Character_Position would be a derived from Standard.Positive, and due to the way Character_Position is derived, as explained above, this would imply the root implementation defined type, should be instead, at least derived from Positive or an ancestor type of Positive.

Which is wrong? TenDRA with Gela-ASIS or the ASIS ISO reference (which I do not own, so I cannot check).

Comments and lightings welcome from any one owning a copy of the ASIS reference, so that I can figure if I should primarily patch Gela-ASIS or AdaControl.

From: Vadim Godunko <vgodunko@gmail.com>
Date: Tue, 9 Nov 2010 06:18:16 -0800 PST
Subject: Re: Asis.Text.Character_Position : inconsistency ?
Newsgroups: comp.lang.ada

[…] Here are official ASIS packages specifications:

http://www.sigada.org/wg/asiswg/specs/asis20s.txt

From: Simon Wright <simon@pushface.org>
Date: Sun, 07 Nov 2010 12:34:56 +0000
Subject: Re: Asis.Text.Character_Position : inconsistency ?
Newsgroups: comp.lang.ada

[…] GNAT ASIS is pretty clear (at least from the copyright notice) that most of the public part of package specs is from the standard.

ASIS.Text is about fragments of code -- the part of the source text corresponding to an element -- in type Span. Line_Number is the line in the source text, Character_Position is the position in the line.

I suspect you're supposed to use Element_Image, Line_Image etc?

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Sun, 07 Nov 2010 20:03:41 +0100
Subject: Re: Asis.Text.Character_Position : inconsistency ?
Newsgroups: comp.lang.ada

[…] Gela-ASIS seems too, as the package specification they used is a raw copy of what you can get from there:

http://www.sigada.org/wg/asiswg/intro.html

> […] I suspect you're supposed to use Element_Image, Line_Image etc?

… and Comment_Image and Non_Comment_Image and Debug_Image, Yes.

But the source I am dealing with makes the assumption that Character_Position is a subtype of Natural, and that is why I was wondering about this type definition. This leads into trouble when I try to build it with Gela-ASIS, because with Gela-ASIS, Character_Position and Line_Number are not subtypes (direct or indirect) of Natural.

As the packages specifications seems to be raw copy of what you get from the link above, I suppose TenDRA is right.

And you are right too, the accessors to be used to get text representation are indeed the Asis.Text.XTZ_Image. So there is no need for Character_Position to be a subtype of the index type of Program_Text. So the ASIS specification is not wrong either.

By the way, another question: do you know a link to a working draft version of ASIS 2005 ?

(I know this can be inferred from ASIS Issues list, just that this would be cleaner… but I don't bother if there is none)

[…]

From: Simon Wright <simon@pushface.org>
Date: Sun, 07 Nov 2010 19:20:40 +0000
Subject: Re: Asis.Text.Character_Position : inconsistency ?
Newsgroups: comp.lang.ada

> the source I am dealing with makes the assumption that Character_Position is a subtype of Natural, and that is why I was wondering about this type definition. […]

I think the source must be making unwarranted assumptions.

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Sun, 07 Nov 2010 21:44:04 +0100
Subject: Re: Asis.Text.Character_Position : inconsistency ?
Newsgroups: comp.lang.ada

[…] But the compiler did not reject anything ;) So the responsibility must be shared with the underlying ASIS library too (remember, Ada is strongly typed, if there was no issue with ASIS-for-GNAT too, this could not even compile with this one)

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 07 Nov 2010 20:35:43 +0100
Subject: Re: Asis.Text.Character_Position : inconsistency ?
Newsgroups: comp.lang.ada

[…] I don't think there is a public version of the current state of the ASIS manual (Randy will correct me if I'm wrong).

Note that there will be no ASIS 2005. Given the current lagging of ASIS w.r.t. Ada, ISO took the decision to skip ASIS 2005 and to issue ASIS 2012, approximately at the same time, and then keep both in sync.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 12 Nov 2010 13:35:16 -0600
Subject: Re: Asis.Text.Character_Position : inconsistency ?
Newsgroups: comp.lang.ada

[…] That's right. The only way to get access to the draft ASIS is to become a member of the ARG. We could use some additional ASIS expertise, so that's not out of the question.

The SIs themselves are publicly available on www.ada-auth.org, so you can find out about individual changes that way.

Ada and Education

Webinar on AdaCore's GPS 5.0

From: Jamie Ayre
Date: Thu, 28 Oct 2010
Subject: GPS 5.0 webinar
Source: LinkedIn Groups - Ada Programming Language

The GNAT Pro InSight webinar series continues with a presentation and demo of the new features introduced in GPS 5.0.

This release sees many enhancements to the GPS IDE technology including improved support for C/C++ in addition to its already comprehensive support for the Ada language, more powerful source editing, improved ease of use, better tool support (GNATstack, CodePeer), and enhanced documentation generation.

This webinar is open to all. If you would like to join us, please visit:

http://www.adacore.com/home/products/gnatpro/webinars/

[webinar to be held on December 14, 2010 - 5:00pm CET —mp]

Webinar on SPARK Pro 9.1

From: AdaCore Webinar webpage
Date: Fri, 12 Nov 2010 [fetched]
Subject: Introducing SPARK 9.1
URL: http://www.adacore.com/home/products/gnatpro/webinars/

The InSight webinar series continues with a presentation by Angela Wallenburg on the new features of the AdaCore/Altran Praxis joint offering – SPARK Pro.

SPARK Pro combines the proven SPARK Ada language and supporting toolset with AdaCore's GNAT Programming Studio (GPS) integrated development environment, backed by unrivalled support systems.

SPARK Pro 9.1 is a major release including many new features – the use of full range array subtypes, the relaxation of aliasing rules for record fields, the ability to specify VC generation on a per-file basis in metafiles, the introduction of new SPARK libraries, the introduction of the SPARKbridge feature.

[webinar to be held on December 7, 2010 - 5:00pm CET —mp]

Ada code examples for educational purposes

From: R Tyler Croy <tyler@linux.com>
Date: 12 Nov 2010 20:25:16 GMT
Subject: Finding code to read for educational purposes
Newsgroups: comp.lang.ada

I've commented on a GitHub support ticket to ask them to support/index Ada code in their "Explore" functionality, but I'm wondering if there are any other good sites to find collections of well commented, well structured Ada code to help me learn Ada properly?

Somewhere between simple one page tutorials and giant programs would be helpful.

From: Jeffrey Carter <jrcarter@acm.org>
Date: Fri, 12 Nov 2010 14:22:56 -0700
Subject: Re: Finding code to read for educational purposes
Newsgroups: comp.lang.ada

[…]

You can find plenty of examples of Ada code at http://www.adaworld.com/ and http://www.adaic.org/. Many of them are libraries, so they're reasonably easy to understand.

The Ada-specific search at AdaIC.org might also be useful.

You can find both the PragmAda Reusable Components and the Mine_Detector game (not giant, but a complete program) at

http://pragmada.x10hosting.com/

From: Thomas Løcke <tl@ada-dk.org>
Date: Sat, 13 Nov 2010 11:14:32 +0100
Subject: Re: Finding code to read for educational purposes
Newsgroups: comp.lang.ada

[…]

I personally enjoy the Ada examples at http://rosettacode.org

http://rosettacode.org/wiki/Category:Ada

Some links from the Ada-DK website:

http://ada-dk.org/?page=basics

http://wiki.ada-dk.org

http://wiki.ada-dk.org/index.php/Ada_Resources

Another option is keeping an eye on http://planet.ada.cx for various Ada projects. Download, unpack, enjoy!

From: Simon Wright <simon@pushface.org>
Date: Sat, 13 Nov 2010 12:27:49 +0000
Subject: Re: Finding code to read for educational purposes
Newsgroups: comp.lang.ada

[…]

If you go to sourceforge.net and search for 'ada' you'll get 138 results -- OK, the first isn't appropriate, but the rest of the first page of results looks relevant, I didn't check further.

From: Georg Bauhaus <rm-host.bauhaus@maps.futureapps.de>
Date: Sat, 13 Nov 2010 14:18:00 +0100
Subject: Re: Finding code to read for educational purposes
Newsgroups: comp.lang.ada

[…]

The GNAT distribution always includes an examples directory.

I think some programs in there will be about a few pages in length.

From: Ed Falis <falis@verizon.net>
Date: Sat, 13 Nov 2010 07:11:23 -0800 PST
Subject: Re: Finding code to read for educational purposes
Newsgroups: comp.lang.ada

[…]

You might find the "Ada Gems" page at AdaCore's site interesting:

http://www.adacore.com/category/developers-center/gems/

These are a set of small tutorials on various topics having to do with Ada - there are 90 some-odd ones there. Also, if you go to libre.adacore.com, sources are available for all of the downloadable tools and components.

From: Marc A. Criley <mc@mckae.com>
Date: Sun, 14 Nov 2010 18:10:00 +0100
Subject: Re: Finding code to read for educational purposes
Newsgroups: comp.lang.ada

The Ada sub-reddit (http://www.reddit.com/r/ada) has links to a variety of Ada related information, including Ada software projects, nearly all of which provide source code.

Ada-related Resources

DSA Messenger

From: Vadim Godunko <vgodunko@gmail.com>
Date: Tue, 26 Oct 2010 15:02:07 -0700 PDT
Subject: Announce: DSA Messenger example
Newsgroups: comp.lang.ada

Hello,

I am pleased to announce new example of use QtAda in distributed application - DSA Messenger. It is simple multiuser chat application, which uses DSA application personality of PolyORB to split Ada application into several partitions and QtAda to implement client's GUI.

You can download source code by following this link:

http://adaforge.qtada.com/cgi-bin/tracker.fcgi/qtada4/downloader/download/file/1

and found important additional information

http://adaforge.qtada.com/cgi-bin/tracker.fcgi/qtada4/wiki/Examples/DSAMessenger

Ada-related Tools

Simple components for Ada v3.10

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 30 Oct 2010 21:45:32 +0200
Subject: ANN: Simple components for Ada v3.10
Newsgroups: comp.lang.ada

The library provides implementations of smart pointers, directed graphs, 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, reentrant 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. The library is kept conform to both Ada 95 and Ada 2005 language standards.

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

The version 3.10 extends the package Object.Handle.Generic_Set and provides some bug fixes.

[see also "Simple Components for Ada v3.9" in AUJ 31-3 (Sep 2010), p.156 —mp]

Fuzzy sets for Ada v5.5

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 31 Oct 2010 10:09:22 +0100
Subject: ANN: Fuzzy sets for Ada v5.5
Newsgroups: comp.lang.ada

The current version includes distributions of string edit, interval arithmetic and simple components packages. It provides implementations of:

1. Confidence factors with the operations not, and, or, xor, +, *;
2. Classical fuzzy sets with the set-theoretic operations and the operations of the possibility theory;
3. Intuitionistic fuzzy sets with the operations on them;
4. Fuzzy logic based on the intuitionistic fuzzy sets and the possibility theory;
5. Fuzzy numbers both integer and floating-point ones with conventional arithmetical operations;
6. Dimensioned fuzzy numbers;
7. Fuzzy linguistic variables and sets of linguistic variables with operations on them;
8. Dimensioned fuzzy linguistic variables and sets;
9. String-oriented I/O is supported;
10. GUI interface based on GTK+ (The GIMP Toolkit) with fuzzy set editors, truth values widgets and renderers, linguistic variables sets editors.

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

Changes to the version 5.4.

- A fuzzy set renderer property was added to prefix the rendered value with a string;
- Fuzzy set renderer commits changes when Enter is pressed while no drop down has been popped up;
- Procedure Get added to Gtk.Generic_Fuzzy_Linguistic_Set_Measure_Tree_View and Gtk.Generic_Fuzzy_Linguistic_Set_Measure_Editor to get dimensioned set of linguistic variables as a dimensionless set and scale;
- Fuzzy.Stream_IO provides conversions to stream elements arrays;
- Minor bug fixes in Fuzzy.Abstract_Edit.Named.

[see also "Fuzzy Sets for Ada v5.4" in AUJ 30-3 (Sep 2009), p.142 —mp]

AVR-Ada 1.1.0 port with AVR-GCC 4.3.2 for OpenBSD

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Wed, 1 Sep 2010 04:00
Subject: AVR-Ada 1.1.0 port with AVR-GCC 4.3.2 for OpenBSD
URL: http://tero.stronglytyped.org/

I updated my AVR-Ada port to version 1.1.0. It consists of three parts:

- AVR-GCC 4.3.2 with Ada support - http://bitbucket.org/tkoskine/avr-gcc
- AVR-Ada 1.1.0 runtime files - http://bitbucket.org/tkoskine/avr-ada-rts
- AVR-Ada 1.1.0 library files (AVR.* packages) - http://bitbucket.org/tkoskine/avr-ada-lib

For now, only Arduino (atmega328p) is supported in avr-ada-lib package.

[see also "AVR-Ada 1.1" in AUJ 31-1 (Mar 2010), p.9 —mp]

Arduino Ethernet Shield support for AVR-Ada

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Wed, 29 Sep 2010 17:53
Subject: Arduino Ethernet Shield support for AVR-Ada
URL: http://tero.stronglytyped.org/

I finally got my code working with Arduino Ethernet Shield and put it available at

http://bitbucket.org/tkoskine/arduino-ethernet/.

Only receiving data via TCP client connections is supported, but I plan to improve the library as my time permits.

[see also http://www.arduino.cc/en/Main/ArduinoEthernetShield —mp]

Mathpaqs - November 2010

From: Gautier de Montmollin <gdemont@users.sourceforge.net>
Date: Wed, 10 Nov 2010 13:11:45 -0800 PST
Subject: Mathpaqs release 10-Nov-2010
Newsgroups: comp.lang.ada

Hello,

There is a new release of mathpaqs, a set of various mathematical packages in Ada including algebra, finite elements, random variables, probability dependency models, unlimited integers.

In this release, the main change is a bug in the Gaussian copula simulation which has been fixed.

http://sf.net/projects/mathpaqs/

Ada Server Faces

From: Stephane Carrez
Date: Thu, 11 Nov 2010 23:21 +0100
Subject: Ada Servlet Example
URL: http://blog.vacs.fr/index.php?post/2010/11/11/Ada-Servlet-Example

To write a web application, Java developers can use the servlet API.

The servlet technology, created around 1997, is a simple and powerful framework on top of which many web applications and higher web frameworks have been created.

This article shows how to write the same kind of web application in Ada.

Ada Servlet Framework

The Ada Servlet framework is provided by Ada Server Faces. It is an adaptation and implementation of the JSR 315 (Java Servlet Specification) for the Ada 05 language.

The Ada API is very close to the Java API as it provides the Servlet, Filter, Request,

Response and Session types with quite the same methods. It should be quite easy for someone who is familiar with Java servlets to write an Ada servlet.

The Ada Servlet implementation uses the Ada Web Server as a web server. In the future other other web servers such as Apache or Lighthttpd could be used.

[download the sources of the project, available under the Apache License 2.0, at http://code.google.com/p/ada-asf/ —mp]

Ada binding for the shapelib library

From: Ian Clifton <cliftons_oxf@yahoo.co.uk>
Date: Fri, 01 Oct 2010 21:29:49 +0100
Subject: Ada shapelib binding?
Newsgroups: comp.lang.ada

Has anyone got an Ada binding to the shapelib library for ESRI shapefiles for geographic data?

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

or any other way of reading shapefiles from Ada. I probably only want to read polygons. I could bite the bullet and cobble together some partial thin routines, I as just wondering if anyone has anything better already. […]

From: Tom Moran <tmoran@acm.org>
Date: Fri, 1 Oct 2010 21:47:59 +0000 UTC
Subject: Re: Ada shapelib binding?
Newsgroups: comp.lang.ada

[…]

I did one several years ago to read BLM data for Scout. It's at:

http://home.comcast.net/~tommoran4/shp.zip

QtAda 3.1.0

From: Vadim Godunko <vgodunko@gmail.com>
Date: Sun, 10 Oct 2010 07:17:50 -0700 PDT
Subject: Announce: QtAda 3.1
Newsgroups: comp.lang.ada

We are pleased to announce the immediate availability of the QtAda 3.1.0. You can download multi platform source code package or Microsoft Windows binary package from the our download page:

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

QtAda is an Ada2005 language bindings to the Qt libraries and a set of development 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 3.1.0:

- support for QtOpenGL module
- support for QDir, QFileInfo, QProcessEnvironment, QSettings, QUrl classes of QtCore module
- support for QAbstractTextDocumentLayout, QColorDialog, QGraphicsObject, QGraphicsSimpleTextItem, QGraphicsTextItem, QImage, QImageReader, QImageWriter, QPlainTextDocumentLayout, QRgb classes of QtGui module
- support for additional operations of classes of QtCore and QtGui modules
- support for mixed GtkAda/QtAda applications on X11 platform
- bug fixes and performance speedups

[see also "QtAda 3.0 and 2.2" in AUJ 30-3 (Sep 2009), p.145 —mp]

Ada bindings for cURL

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Wed, 13 Oct 2010 19:23
Subject: Ada bindings for cURL
URL: http://tero.stronglytyped.org/

I put my Ada bindings to libcurl available at http://hg.stronglytyped.org/curl-ada/.

At the moment, they are pretty simple and contain only a small subset of libcurl, but they allow me to fetch data over http/https and that is good enough for my current purposes.

The bindings should work with GNAT and Janus/Ada on 32-bit and 64-bit systems.

The build scripts are less than optimal, but with some effort you should figure out how to build the bindings.

PC/SC WinSCard API

From: Ludovic Rousseau's blog
Date: Fri, 20 Aug 2010 05:32 CEST
Subject: PCSC sample in Ada
URL: http://ludovicrousseau.blogspot.com/2010/08/pcsc-sample-in-ada.html

Here is the PC/SC [WinSCard —mp] sample in Ada language I promised […].

Installation

The PCSC/Ada project is hosted at http://www.nongnu.org/pcscada.

PCSC/Ada is available as package in Debian testing/unstable and Ubuntu 10.04.

The API documentation is available online at

http://www.nongnu.org/pcscada/api/index.html.

[…]

I do not use Ada myself so I can't really say more. This wrapper should do the job if you do use Ada.

Thanks a lot to Reto Buerki, the author of the Ada wrapper, for writing the sample code for me.

P2Ada

From: Gautier de Montmollin <gdemont@users.sourceforge.net>
Date: Mon, 6 Sep 2010 14:34:58 -0700 PDT
Subject: ANN: P2Ada August 2010
Newsgroups: comp.lang.ada

Hello.

Source codes and now binaries of P2Ada for Windows and Mac PPC / Intel are available on Source Forge web (Aug-2010):

http://sourceforge.net/projects/p2ada/files/

P2Ada is a set of helpful tools used to translate Pascal source code in Ada source code.

P2Ada includes :

- aflex lexer
- ayacc parser
- NewP2Ada translator for ISO standards
- ObjP2Ada translator for Delphi and FPC

NewP2Ada translator is well known and stable.

ObjP2Ada translator is new with a very preliminary release.

Feel free to send feedback on web site:

http://sourceforge.net/projects/p2ada/support

[…]

libsparkcrypto 0.1.0

From: Alexander Senier <mail@senier.net>
Date: Thu, 9 Sep 2010 00:08:53 +0200
Subject: ANN: libsparkcrypto
Newsgroups: comp.lang.ada

Hello,

I'm happy to announce the first release of libsparkcrypto.

Further information, license and source code is available here:

http://senier.net/libsparkcrypto/

Version 0.1.0 of libsparkcrypto can be downloaded here:

http://senier.net/libsparkcrypto/libsparkcrypto-0.1.0.tgz

libsparkcrypto is a formally verified implementation of several widely used symmetric cryptographic algorithms using the SPARK programming language and toolset. For the complete library proofs of the absence of run-time errors like type range violations, division by zero and numerical overflows are available. Some of its subprograms include proofs of partial correctness.

The distribution contains test cases for all implemented algorithms and a benchmark to compare its performance with the OpenSSL library. The achieved speed has been found to be very close to the optimized C and Assembler implementations of OpenSSL.

[…]

ssprep-1.5.3

From: Per Sandberg <per.sandberg@bredband.net>
Date: Wed, 13 Oct 2010 17:11:02 +0200
Subject: ANN: ssprep-1.5.3
Newsgroups: comp.lang.ada

ssprep contains a few helpers to manage and generate projects based on GNAT project files.

ssprep: which is a tool to generate project structures from templates using the templates_parser.

getbuildorder: Which is a tool that takes a set of GNAT project files and calculates the build order it could be used to automatically perform a make/make install in correct order.

From: Per Sandberg <per.sandberg@bredband.net>
Date: Wed, 13 Oct 2010 17:51:53 +0200
Subject: Re: ANN: ssprep-1.5.3
Newsgroups: comp.lang.ada

Sorry missed the link:

http://sourceforge.net/projects/ssprep/

[…]

TOMI_4_Ada 0.10

From: Marc A. Criley <mc@mckae.com>
Date: Mon, 25 Oct 2010 19:13:12 -0500
Subject: Announce: TOMI_4_Ada 0.10 Initial Release
Newsgroups: comp.lang.ada

TOMI_4_Ada is "Text-Oriented Messaging Interfaces 'for' Ada", providing a basic Ada interface to existing messaging middleware providers, and a framework for expanding support to currently unsupported ones. While most messaging middleware supports binary as well as text messaging (text being simply a subset of binary), TOMI_4_Ada focuses exclusively on textual transfers so as to simplify the interface, but remaining well-equipped to support the broad domain of text-based messaging protocols, i.e. XML, HTML, JSON, raw text, etc.

Out of the box TOMI_4_Ada provides interfaces to the text messaging capabilities of three protocols:

- STOMP protocol via the Apache ActiveMQ broker
- AMQP via the OpenAMQ broker
- ZeroMQ (brokerless)

TOMI_4_Ada provides the following capabilities:

- A "thick" binding to a subset of each of the supported messaging protocols that includes a callback-based publish/subscribe and client/server reference implementation.
- A "thin" binding to a subset of the WireAPI protocol that is supplied as part of the OpenAMQ message broker distribution.
- The STOMP protocol is simply a definition of a text-based protocol, and the TOMI_4_Ada.STOMP_Adapter package implements it. An externally supplied transport protocol (such as TOMI_4_Ada.Transport.TCP_Connect) and supporting messaging broker, such as Apache ActiveMQ, is then required to convey messages utilizing the STOMP protocol.

In addition a number of simple test programs are provided in the 'test' directory that exercise various aspects of the thick bindings. These can be used as a starting point for implementing one's own application-specific messaging.

TOMI_4_Ada was developed on Linux (Ubuntu 10.04) using the GNAT GPL 2010 compiler and verified in that environment with the Apache ActiveMQ broker, the OpenAMQ broker, and Per Sandberg's ZeroMQ binding.

TOMI_4_Ada 0.10 is considered an alpha release, but currently has no known outstanding bugs. It is available on SourceForge at

http://sourceforge.net/projects/tomi4ada/files.

[…]

Fuzzy machine learning framework v0.1

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 7 Nov 2010 10:42:05 +0100
Subject: ANN: Fuzzy machine learning framework v0.1
Newsgroups: comp.lang.ada

Fuzzy machine learning framework is a library and a GUI front-end for machine learning using intuitionistic fuzzy data. The approach is based on the intuitionistic fuzzy sets and the possibility theory. Further characteristics are: fuzzy features and classes; numeric, enumeration features and features based on linguistic variables; user-defined features; derived and evaluated features; classifiers as features for building hierarchical systems; automatic refinement in case of dependent features; incremental learning; fuzzy control language support; object-oriented software design with extensible objects and automatic garbage collection; generic data base support through ODBC; text I/O and HTML output; advanced graphical user interface based on GTK+; examples of use.

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

Ada-related Products

AdaCore — GNATbench 2.4.1 for Wind River Workbench

From: AdaCore Press Center
Date: Tue, 21 Sep 2010
Subject: AdaCore Upgrades GNATbench for Wind River Workbench
URL: http://www.adacore.com/2010/09/21/gnatbench-241/

New platforms and features benefit Ada developers on Wind River Workbench

NEW YORK and PARIS, September 21, 2010 – Embedded Systems Conference Boston […]

AdaCore, a leading supplier of Ada development tools and support services, today announced a major update to GNATbench, the Ada plug-in that brings the advantages of AdaCore's GNAT Pro toolset to Wind River's Workbench integrated development environment for embedded systems running VxWorks. The latest version of GNATbench, 2.4.1, is now available as part of the recent GNAT Pro 6.3.2 release, and is supported on Wind River Workbench 3.1 / 3.2, hosted on Windows, Linux, and Solaris.

"AdaCore is one of Wind River's strategic partners, and our business relationship goes back many years," said Robert Dewar, AdaCore President and CEO. "Maintaining this relationship is an ongoing process, it involves making critical contributions to our joint technology and enhancing our products to meet customers' growing needs.

Our new version of GNATbench meets both these goals."

"We've introduced support for significant new Wind River platforms as well as for a new version of an existing major platform," said Dr. Patrick Rogers, GNATbench Project Lead. "These platforms address the application domains within which Ada is most advantageous. We've also enhanced our integration with the Wind River project builder and replaced the underlying Ada compilation engine."

GNATbench 2.4.1 now supports three new Wind River platforms: VxWorks 653, version 2.3; VxWorks Cert, version 6.6.2; and VxWorks MILS, version 2.1.

In addition, GNATbench now supports VxWorks 6.8 (General Purpose Platform 3.8) and Workbench 3.2. Tutorials for these new platforms are now provided, as well as new tutorials for building shared libraries.

GNATbench now uses the Wind River default "flexible" managed build type for

all new projects. This build type allows only those directories actually containing code to be processed during a build, which is therefore simpler and more efficient.

The deprecated "standard" managed builds are still supported for existing GNATbench projects, and a project conversion wizard provides the option to convert them to use the new build type.

This new version of GNATbench uses GPRbuild, the AdaCore multi-language builder, when building the Ada portion of a Workbench project. As such, it can also compile code written in other languages, if required. Equally important, GPRbuild is the current and future AdaCore builder technology, replacing the older gnatmake toolset.

"Wind River adds new features and functionality to each new version of Workbench based on customer demand," said Chip Downing, Director of Aerospace and Defense at Wind River. "With AdaCore's continued enhancements to GNATbench, the Workbench/GNATbench combination provides a very powerful Eclipse-based development environment."

New GNATbench 2.4.1 features include:

- New Wind River platforms
- Wind River VxWorks 653 Platform, version 2.3
- Wind River VxWorks DO-178B Platform, version 6.6
- Wind River VxWorks MILS Platform, version 2.0.1
- New version of Wind River General Purpose Platform
- Workbench 3.2
- VxWorks 6.8
- "Flexible" managed builds for all new projects
- Project conversion wizard
- GPRbuild, the GNAT multi-language builder
- New tutorials
- VxWorks 653
- VxWorks DO-178B
- VxWorks MILS
- Shared libraries

About GNATbench

GNATbench for Wind River Workbench brings the advantages of AdaCore's GNAT Pro toolset to Wind River's Workbench integrated development environment for embedded systems running VxWorks. GNATbench for Wind River Workbench is fully integrated with the existing Workbench tools, combining the power of AdaCore's development and compilation technology with the extensive Workbench tools for VxWorks systems creation.

AdaCore has also developed a stand-alone version of GNATbench for Eclipse. This separate plug-in has all the editing and browsing features of the Workbench version, including the Outline View. The difference is primarily the intended execution target: the builder produces executables for native systems, rather than embedded processors, and likewise the debugger supports native system debugging.

About AdaCore

Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, a state-of-the-art programming language designed for large, long-lived applications where safety, security, and reliability are critical. AdaCore's flagship product is the GNAT Pro development environment, which comes with expert on-line support and is available on more platforms than any other Ada technology. AdaCore has an extensive world-wide customer base; see http://www.adacore.com/home/company/customers/ for further information.

Ada and GNAT Pro continue to see a growing usage in high-integrity and safety-certified applications, including commercial aircraft avionics, military systems, air traffic management/control, railroad systems, and medical devices, and in security-sensitive domains such as financial services.

AdaCore has North American headquarters in New York and European headquarters in Paris.

www.adacore.com

AdaCore — GNAT Pro 6.3 Multi-Language

From: AdaCore Press Center
Date: Tue, 21 Sep 2010
Subject: Enhanced Solutions for Multi-Language Systems
URL: http://www.adacore.com/2010/09/21/multi-language/

NEW YORK and PARIS, September 21, 2010 – Embedded Systems Conference Boston AdaCore, a leading supplier of Ada development tools and support services, today announced a comprehensive set of tools and support services for projects where Ada is used in conjunction with other programming languages. Available with GNAT Pro 6.3, the latest release of AdaCore's Ada Development Environment, the solutions include tools and libraries to handle the various ways in which multi-language systems are designed and constructed.

"'One Language Fits All' is not how large systems are developed," said Robert Dewar, AdaCore President and CEO. "Programmers need to mix and match, using different languages that are appropriate for different jobs, or incorporating legacy software components written in different languages. AdaCore is answering that requirement, through both products and support services, for customers who are using other languages along with Ada."

AdaCore's multi-language solutions include GNAT Pro C and GNAT Pro C++ for support of C and C++ development, respectively, as well as a general-purpose multi-language build tool (GPRbuild). For systems that need to work with Java, AdaCore supplies the GNAT Ada-Java Interfacing Suite (for communicating between Java and natively compiled Ada) and GNAT Pro for the JVM. Combining Ada and Python, for example to drive Ada test suites through Python scripts, is supported by the GNAT Component Collection (GNATcoll). And GNAT Pro for .NET allows smooth interfacing, through managed code, between Ada and C# or other languages that compile to Common Language Runtime assemblies. These are in addition to GNAT Pro's existing support for the foreign language interfacing facilities specified in the Ada standard.

Multi-language capabilities are especially important in safety-critical and/or high-security applications. For example avionics systems typically consist of components at different safety levels – such as flight software at DO-178B Level A in Ada, and entertainment software at Level E, perhaps in Java.

In the security arena, a MILS-compliant architecture can host different applications at different Evaluation Assurance Levels (EALs), where applications at the highest levels might be written in a language such as SPARK (an Ada subset augmented with annotations / "contracts" that allow formal proofs of security properties), whereas applications at lower levels might be written in languages that do not support such rigor. AdaCore's multi-language solutions address such needs.

About GNAT Pro The GNAT Pro development environment is a full-featured, multi-language development environment complete with libraries, bindings and a range of supplementary tools. It provides a natural solution for organizations that need to create reliable, efficient, and maintainable code.

GNAT Pro implements all three versions of the Ada language standard – Ada 83, Ada 95, and Ada 2005 – and the latest releases of GNAT Pro implement some of the new features in Ada 2012. GNAT Pro is backed by rapid and expert support service.

[…]

AdaCore — GNAT GPL for the LEGO MINDSTORMS NXT - Ravenscar Edition

From: Matteo Bordin <matteo.bordin@gmail.com>
Date: Thu, 21 Oct 2010 08:39:32 -0700 PDT
Subject: GNAT GPL for the LEGO MINDSTORMS NXT - Ravenscar Edition
Newsgroups: comp.lang.ada

Dear group,

AdaCore is proud to announce the availability of the GNAT GPL edition for the LEGO MINDSTORMS NXT - Ravenscar Edition. This new release of the GNAT GPL for the LEGO MINDSTORMS NXT includes a run-time library that supports the Ada 2005 Ravenscar profile. With the Ravenscar run-time library, developers can write analyzable and efficient all-Ada concurrent real-time applications on the LEGO MINDSTORMS NXT. The compiler implements the Ada 2005 standard and provides a preview of some Ada 2012 features. The release includes Ada drivers to access the NXT brick and its connected sensors and actuators. This technology is community-based: users are encouraged to contribute additional drivers, teaching material and demos.

Have a look at

http://libre.adacore.com/libre/tools/mindstorms/

Have fun!

[…]

[see also "AdaCore — GNAT GPL for Lego Mindstorms NXT" in AUJ 30-3 (Sep 2009), p.148 —mp]

AdaCore — GPS 5.0

From: AdaCore Press Center
Date: Tue, 26 Oct 2010
Subject: AdaCore releases GPS 5.0
URL: http://www.adacore.com/2010/10/26/gps-5-0/

GPS 5.0 Integrated Development Environment brings enhanced multi-language support, more powerful source editing, and improved ease of use.

NEW YORK, PARIS and FAIRFAX, Va., October 26, 2010 – SIGAda 2010 – AdaCore, a leading supplier of Ada language tools and support services, today announced the release of GNAT Programming Studio (GPS) 5.0. This new major version of AdaCore's graphical Integrated Development Environment (IDE) offers enhanced support for C and C++, more powerful source editing, simpler use, and integration of GNATstack (a static analysis tool that determines a program's maximum stack requirements). GPS is provided with GNAT Pro on most platforms, for both native and embedded software development.

GPS's multi-language support will especially benefit GNAT Pro customers whose applications include C or C++ as well as Ada. Among the enhancements are more accurate and complete source navigation using a new cross reference engine, better outlining and indentation, and navigation through #include directives.

GPS 5.0 also brings easier source editing via additional syntax highlighting, annotations on the side of editor windows concerning compilation messages and search results, automatic compilation, highlighting of errors, improved code completion, and better automated code fixes. It also introduces easier target toolchain selection, support for GNATstack, and access to project templates for easy project setup.

"Developers will enjoy this major release, with the many powerful new features it places at their fingertips," said Arnaud Charlet, GPS Project Manager at AdaCore. "GPS 5.0 is really a must for multi-language projects – it uses an upgraded technology that we've been developing during the past few years."

Enhancements in GPS 5.0 include:

- Improved support for C/C++:
  - o More accurate and complete source navigation
  - o Better outline view
  - o Improved automatic indentation
- More powerful source editing:
  - o More syntax highlighting
  - o Annotations on the side of the editor window
  - o Automatic compilation
  - o Enhanced code completion
  - o Partial Ada source navigation without compilation
  - o Additional automatic code fixes
- Improved ease of use:
  - o Easy target toolchain selection
  - o Faster processing on large projects
  - o Improved handling of desktop via perspectives
  - o Ability to quickly create projects from existing templates
- Better tool support:
  - o Support for GNATstack
  - o Improved support for CodePeer
- Enhanced documentation generation:
  - o Detection of entity names in comments and production of links to their definitions
  - o Handling of lists and intentional line returns in structured comments

GPS 5.0 is compatible with GNAT Pro versions 3.16a1 up to 6.4.

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

About GNAT Programming Studio (GPS)

GPS is a powerful Integrated Development Environment (IDE) written in Ada using the GtkAda toolkit. GPS's extensive source-code navigation and analysis tools can generate a broad range of useful information, including call graphs, source dependencies, project organization, and complexity metrics. It also supports configuration management through an interface to third-party Version Control Systems, and is available on a variety of platforms. GPS is highly extensible; a simple scripting approach enables additional tool integration. It is also customizable, allowing programmers to specialize various aspects of the program's appearance in the editor for a user-specified look and feel.

[…]

Inspirel — YAMI4 v. 1.1.0 and 1.2.0

From: Maciej Sobczak <maciej@msobczak.com>
Date: Wed, 1 Sep 2010 05:28:00 -0700 PDT
Subject: YAMI4 v. 1.1.0 released
Newsgroups: comp.lang.ada

I am pleased to announce that the new version of YAMI4, 1.1.0, has been just released and is available for download.

The Ada programmer will benefit from this new release thanks to the ability to transmit raw binary messages - this in addition to support high-performance scenarios can be used as a hook for custom serialization routines. Direct improvements for Ada include also better control over the automatic reconnection facility and reduced jitter in multiple-receiver scenarios, even in the case of partial system failure.

This new version extends the coverage of supported programming languages with a completely new Python3 module, which features full integration of built-in dictionary objects as message payloads.

Since Python is frequently being used as a secondary language in the Ada community, the support for it will allow Ada programmers to better integrate their components in multi-language distributed systems.

Please see the changelog.txt file, which is part of the whole package, for a detailed description of all improvements.

http://www.inspirel.com/yami4/

From: Maciej Sobczak <maciej@msobczak.com>
Date: Mon, 8 Nov 2010 03:00:52 -0800 PST

Subject: YAMI4 v. 1.2.0 released
Newsgroups: comp.lang.ada

Hello,

I am pleased to announce that the new version of YAMI4, 1.2.0, has been just released and is available for download:

http://www.inspirel.com/yami4/

This release brings the following improvements for Ada programmers:

- Keepalive option for TCP/IP.
- Propagation of events related to connection management.
- Ability to customize the management of slow (overflowing) receivers in the publish-subscribe scenario.

Apart from these, a pending issue related to some combination of compiler bugs was fixed to ensure that the code compiles and runs properly on a wider range of compiler versions.

Multi-language developers will also find it interesting that in addition to equivalent feature improvements for other supported languages, a new module for Python 2.5+ was added to complement the already existing Python 3.x library.

Please see the changelog.txt file for the detailed list of improvements.

[see also "Inspirel — YAMI4" in AUJ 31-2 (Jun 2010), p.91 —mp]

Lattix — Lattix 6.0

From: Lattix Website
Date: Wed, 1 Sep 2010
Subject: Lattix Releases Lattix 6.0 with new Repository and Project Browser
URL: http://www.lattix.com/node/155

Award-winning software architecture management solution now provides powerful way to publish and track the evolution of projects

Boston, MA - September 1, 2010- Lattix Inc., a leading provider of innovative software architecture management solutions, today announced the release of its newest solution, Lattix 6.0. This solution includes the Lattix Repository and Project Browser, a web application that enables architects, developers and managers to view a project's architecture, dependencies, and metrics as well as changes and trends over time.

In addition to many performance and feature improvements, Lattix 6.0 also includes a new ActionScript Module and enhanced capabilities to create projects for entire enterprise systems consisting of requirements, processes, and resources in addition to software and hardware infrastructure.

"With these innovations in Lattix 6.0, the architecture of software becomes more visible to the entire organization" explains Neeraj Sangal, president and founder of Lattix. "This leads to significant improvements in software quality and productivity and helps to make the software development process far more transparent than ever before."

The Lattix Repository and Project Browser allows users to maintain a complete history of their projects. The Lattix Repository, which can be updated manually with Lattix LDM or automatically with Lattix LDC, includes a web server to enable access via a web browser. The extended team can use the Project Browser to view Project Tracks with Snapshots of each build and compare trends of changes, architectural violations, metrics and a variety of other data. It is now easier than ever to communicate this critical information to the entire organization.

For more information about the Lattix Repository and Project Browser, please visit

http://www.lattix.com/repository.

The Lattix ActionScript Module is the most comprehensive solution for analyzing the architecture of your ActionScript and Flex MXML applications. You can load in swf,swc, or link report files and understand the interrelationships between them at any level (packages, classes, interfaces, methods, data members etc.)

"As complexity creeps into client side applications, it has become increasingly important to understand and manage their architecture too," said Han van Roosmalen, Software Architect at Software-Architectuur.nl. "With the Lattix ActionScript Module I was able to find structural problems in my outsourced app development before it was deployed, which saved us a lot of hassle."

For more information at the Lattix ActionScript Module, please visit

http://www.lattix.com/products/actionscript.

About Lattix 6.0

Lattix 6.0 provides the most comprehensive solution for systems that include codebases, databases, frameworks, and UML/SysML models. Lattix 6.0 now supports 64-bit operating systems and has modules for ActionScript, Ada, C/C++, Java, .NET, and Pascal languages; Oracle, SQL Server, and Sybase databases; Spring and Hibernate frameworks; and XMI and IBM Rational Rhapsody models. With the new Repository and Project Browser, Lattix 6.0 also provides support for full web-based viewing of architecture diagrams, reports of key metrics and architectural violations, and trends of snapshots of the project over time.

To learn more about Lattix 6.0 and explore the different solutions that are available, please visit http://www.lattix.com/products.

Lattix 6.0 enables companies to improve and maintain quality, lower defect rates, enhance testability, lower costs through more effective development, and manage risks by better understanding of the impact of proposed changes.

Availability

Lattix 6.0 is available immediately from Lattix in North America or from our Partners throughout Europe, the Middle East, Asia Pacific, and South America. A variety of license options are available, from individual user to enterprise floating licenses. A free evaluation license is also available for download from

http://www.lattix.com/download.

About Lattix

Lattix is the leader of software architecture management solutions that deliver higher software quality and lower risk throughout the application lifecycle.

Lattix provides a powerful new approach which utilizes the Dependency Structure Matrix (DSM) for automated analysis and enforcement of system architectures.

Lattix is located in North Reading, MA. More information about Lattix can be found at

www.lattix.com.

[…]

[see also "Lattix — Lattix 5.5" in AUJ 30-4 (Dec 2009), p.211 —mp]

Ada and GNU/Linux

On porting GNAT JVM/AVR/Mindstorms to GNU/Linux

From: deadlyhead <deadlyhead@gmail.com>
Date: Tue, 26 Oct 2010 17:59:26 -0700 PDT
Subject: GNAT JVM/AVR/Mindstorms on GNU/Linux?
Newsgroups: comp.lang.ada

Just wondering if anybody has had any success using GNAT targeting either JVM, AVR-elf or Mindstorms NXT from a GNU/Linux environment. I've been trying to build a JVM cross-compiler on my Debian system with the latest stable GCC release, but have met only with failure.

Comparing the patches provided by AdaCore and the GCC development tree, it looks like I might have success when GCC 4.6 is released, but until then I think I'm stuck.

I'd really like to develop for the NXT system, too. I have some friends whose kids who would be pretty excited at having robots that could be controlled from their computers, and at least a couple of them are old enough to be

introduced to some basic programming, robotics and computer science geekery. I've been introducing them all to the joys of free software (using GNU/Linux and the Gimp, SuperTuxKart, etc), and it seems a shame that I can't use the same system to teach them about Ada and robotics at the same time.

I'm just fishing for ideas here, and would like to know if anybody has tried/had any success with targeting embedded systems with GNAT from GNU/Linux (I include the JVM here because I'm looking at the possibility for developing Android apps in Ada, too). Am I crazy to think that a GCC-based compiler should work on GNU systems first, then other systems later?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 27 Oct 2010 00:25:17 -0700 PDT
Subject: Re: GNAT JVM/AVR/Mindstorms on GNU/Linux?
Newsgroups: comp.lang.ada

[…]

If you really need GCC 4.6, then I suggest you use it :) You don't have to wait for another year for the release of 4.6.0; you can compile from the sources any day. OK, you might run into bugs but your reports would then be very valuable; the fixes would make it into 4.6.0. Note that the end of Stage 1 is scheduled for today; from then on the sources of GCC 4.6 will be quite stable (no new features, bug fixes only).

> I'd really like to develop for the NXT system, too. […]

FTR, I'd be interested too if I had more time on my hands. My son is only 5, so it's a bit too soon for him anyway. It would be great if Debian would include a cross-compiler for the NXT, nicely packaged and ready to go…

From: deadlyhead <deadlyhead@gmail.com>
Date: Wed, 27 Oct 2010 08:26:03 -0700 PDT
Subject: Re: GNAT JVM/AVR/Mindstorms on GNU/Linux?
Newsgroups: comp.lang.ada

[…]

I've been eyeballing compiling from the GCC trunk for about a week now, and you're right, there is no need for me to wait for a 4.6 release. I used the 4.4 SVN regularly when it was being developed, and never actually had any problems compiling Ada programs. Don't know why I've been hesitant this time.

> […] It would be great if Debian would include a cross-compiler for the NXT, nicely packaged and ready to go… […]

Is this a subtle hint? If so, I'm willing to give it a shot once I'm done moving (I'm wasting precious packing time replying now!) and I'll let you know if I have any success.

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 27 Oct 2010 12:36:22 +0200
Subject: Re: GNAT JVM/AVR/Mindstorms on GNU/Linux?
Newsgroups: comp.lang.ada

[…]

> Just wondering if anybody has had any success using GNAT targeting either JVM, AVR-elf or Mindstorms NXT from a GNU/Linux environment. […]

I am currently at the SIGAda conference, and we had yesterday a paper from the people of Universidad Politecnica de Madrid, who did the port.

I'll transmit your e-mail to Peter Bradley who did the talk.

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 27 Oct 2010 17:46:31 +0200
Subject: Re: GNAT JVM/AVR/Mindstorms on GNU/Linux?
Newsgroups: comp.lang.ada

[…]

> That would be excellent. I wish I could have seen their talk. None of the SIGAda talks are posted online anywhere, are they?

They are not taped, but the slides are generally on the post-conference CD.

BTW, the talk was really about using the Lego-Mindstorms, not about the port to Linux.

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 27 Oct 2010 23:28:06 +0200
Subject: Re: GNAT JVM/AVR/Mindstorms on GNU/Linux?
Newsgroups: comp.lang.ada

[…]

> That would be interesting reading. How do I obtain such a CD?

You normally get it as part of your subscription to SIGAda.

They announced at the conference it would be available early 2011.

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 developer

Development (& designer tests) of basic SW, signalling application software and monitor software (interface Basic Software / Application Software).

Integration. Functional tests in lab.

Responsibilities:

Responsible of SW development (design, integration and designer tests).

Experience:

Experienced in SW design and programming (4 or 5 years).

Experienced in real time embedded Sw and basic Sw.

Notions in communication, safety protocols.

[…]

Technical Skills & Competences

Ada programming language.

Functional SW specification (Teamwork tool).[…]

Job offer [United Kingdom]: Software Engineer

[…] Software Engineer with strong defence background […]

This role is for an experienced and tenacious software engineer with strong ADA - 95 [sic —mp] background. You will work within the engineering team supporting design, development, analysis, testing, and documentation of a complex mission computing system in support of the MCSP.

The programme includes technology enhancements to the aircraft, ground preparation/analysis system, training systems and support infrastructure. […]

The position requires strong understanding and sound application of the Software Engineering principles and practices and a general knowledge of Systems engineering and Test & Integration and Validation disciplines.

The position requires a software developer who designs, develops, documents, tests, and debugs applications software that contains logical and mathematical solutions to business/mission problems or questions in computer language for solutions by means of data processing equipment. Applies the appropriate standards, processes, procedures, and tools throughout the development life cycle.

Corrects program errors, prepares operating instructions, compiles documentation of program development, and analyzes system capabilities to resolve questions of program intent, output requirements, input data acquisition, programming techniques, and controls.

The Software Developer will be responsible for design, development and unit test of elements of software on the MCSP programme. This will include:

- Plan own work within defined constraints with the assistance of engineering and technical mentors
- Provide overall technical support and assistance to engineers

- Design using object oriented methodologies and a UML toolset (Artisan Studio)
- Code development in ADA 95 (using Greenhill's AdaMulti)
- Documentation of developed software
- Unit testing (using ADATest)
- Problem determination and resolution

[…]

Required Skills:

- Ada Programming Experience
- Development with C/C++/Java
- Be familiar with object-oriented design methodologies and UML.
- Experience with Version Manager or equivalent
- Experience with Visual Studio and C# or VB.Net
- Experience with XML, SQL, ODBC, web services
- Experience in developing media-rich GUIs containing audio, video, animation
- Experience in database programming and building web applications with database connectivity

[…]

Desired Skills:

- Development in ADA95
- Familiarity with software safety standards including Def Stan 00-55 and DO178
- Experience of Configuration Management Systems within a Software Development environment
- Experience in OpenGL
- Demonstrable experience of Software Development within a Windows Based System
- Experience with iData or equivalent (i.e. VAPS)
- CORBA
- Web services, XML, SQL/Oracle
- DOORS experience

Job offer [United Kingdom]: Principal Software Engineer

Principal Software Engineer with Avionics, Defence background required to work within a world leading Defence Company.

As a Principal Software Engineer, you will have experience within a software lead or senior role.

You will have software engineering experience in embedded software systems in safety critical applications in the Aerospace, Defence, transport or space domains.

You will have experience in full lifecycle development, from system level requirements analysis and failure moding through functional allocation, modelling, and interface definition to software development and multi-level verification.

- Experience in software technical authority/lead
- Requirements analysis
- UML Design
- DO-178B/ED-12B/CENELEC BS EN50128

Experience of waterfall, incremental and iterative development programmes, prototyping, customised V-lifecycles and AGILE SCRUM AND XP practices.

Desired Skills:

- UML
- Simulink
- SCADE
- Ada
- C
- C++
- DOORS or Requisite pro
- Clear case, CM Synergy, Subversion, Tortoise SVN or PVCS/Dimensions.
- Artisan or Rational Rose/Rhapsody
- ARINC653/429
- Canbus

Job offer [Italy]: Software Engineer

Software Engineer – Design, Development and Testing of avionics systems

At least 6 months of experience in the avionics domain is required.

The knowledge of the following programming languages, tools and methodologies is required:

- Experience of programming in ADA [sic —mp] for embedded applications, preferably in the avionics domain;
- Software Testing;
- C/C++;
- DOORS, Matlab, Vector.

[…]

The knowledge of avionics standards such as DO-178B/254 etc. is fundamental.

The candidate must hold a Laurea [Bachelor's degree —mp] in Computer, Electronic or Telecommunication Engineering.

[translated from Italian —mp]

Job offer [Spain]: Analyst programmer

The job assignment comprises:

- Training period on railway signalling systems;
- Programming with low-level languages;
- Design and development of test cases for hardware simulator;

[…]

No experience is required.

- Ingeniero Superior [~ Master's degree —mp] in Telecommunication Engineering;
- Good knowledge of network programming;
- Good knowledge of languages for multitasking (C/C++ and ADA [sic —mp]).

[…]

- A master's thesis realized with C++ is considered an advantage.

[translated from Spanish —mp]

Ada in Context

On the Ravenscar profile and task termination

From: Georg Bauhaus <rm-host.bauhaus@maps.futureapps.de>
Date: Fri, 05 Nov 2010 12:05:42 +0100
Subject: Q: Profiles
Newsgroups: comp.lang.ada

The Ravenscar profile (and the Restricted profile in GNAT) assume we want our tasks to run forever. Programs will be supported by a lean and efficient run-time.

What if I just want the lean and efficient Ravenscar run-time but do want my tasks to terminate?

From: Georg Bauhaus <rm-host.bauhaus@maps.futureapps.de>
Date: Fri, 05 Nov 2010 12:50:50 +0100
Subject: Re: Q: Profiles
Newsgroups: comp.lang.ada

> […]

> What is the impact of using a system designed to run for ever for an application which is expected to run short or not very long ?

Part of the system (run-time system?) is that it is lean and efficient.

Assume programs that are designed to run for a few days.

Their communication structure is easily expressed since the tasks communicate along very few lines.

No fancy tasking things (Ravenscar is on).

Basically, the tasks express independent sequences of statement whose results is to be coordinated every once in a while, and when they have finished.

From: Vinzent Hoefler <nntp-2010-10@t-domaingrabbing.de>
Date: Fri, 05 Nov 2010 21:14:58 +0100
Subject: Re: Q: Profiles
Newsgroups: comp.lang.ada

[…]

I'd suggest to use the appropriate list of Restriction pragmas then.

See ARM D13.1(3) f.

From: Georg Bauhaus <rm-host.bauhaus@maps.futureapps.de>
Date: Fri, 05 Nov 2010 21:59:00 +0100
Subject: Re: Q: Profiles
Newsgroups: comp.lang.ada

[…]

I think I need to place a pragma Profile (Ravenscar) or (Restricted) (in the case of GNAT) to make the compiler pick the desired run-time. But then tasks won't terminate.

From: Vinzent Hoefler <nntp-2010-10@t-domaingrabbing.de>
Date: Fri, 05 Nov 2010 22:28:47 +0100
Subject: Re: Q: Profiles
Newsgroups: comp.lang.ada

[…]

If you can stick to GNAT only, this isn't true. "No_Task_Termination" is an additional restriction for the Ravenscar profile which isn't in the original set. See GNAT RM:

> The above set is a superset of the restrictions provided by pragma Profile (Restricted), |it includes six additional restrictions (Simple_Barriers, No_Select_Statements, No_Calendar, No_Implicit_Heap_Allocations, No_Relative_Delay and *No_Task_Termination*).
>
> This means that pragma Profile (Ravenscar), like the pragma Profile (Restricted), automatically causes the use of a simplified, more efficient version of the tasking run-time system.

Maybe I am wrong, but I would expect the binder to use the "simplified, more efficient version" as soon as the proper set of restrictions is met, no matter if they are given in a "pragma Profile (…)" or as an explicit list of "pragma Restrictions (…)".

And BTW, the "tasks won't terminate" is probably not true in its literal sense, it's just erroneous behaviour if they do in a Ravenscar profile restricted program. ;)

From: Ed Falis <falis@verizon.net>
Date: Fri, 5 Nov 2010 16:11:37 -0700 PDT
Subject: Re: Q: Profiles
Newsgroups: comp.lang.ada

The --RTS= switch for gnatmake or --config=,,,, switch to gprconfig is your friend. You generally would need to designate the run-time library through one of these mechanisms. pragma Profile (Ravenscar) will remove some overhead, but not as much as the former in versions of GNAT that support the specialized libraries.

As far as termination goes, it's erroneous as you stated, but there are a variety of unusual ways to terminate tasks, typically being system calls.

From: Ed Falis <falis@verizon.net>
Date: Sat, 6 Nov 2010 07:43:55 -0700 PDT
Subject: Re: Q: Profiles
Newsgroups: comp.lang.ada

> The --RTS=… option is really useful with GNAT-Pro, isn't it ?

Right, the GNAT Pro line is the place supporting alternate run-time libraries of the various GNATs.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sat, 06 Nov 2010 15:53:16 +0100
Subject: Re: Q: Profiles
Newsgroups: comp.lang.ada

[…]

No, the gnat-4.4 in Debian also supports both SJLJ and ZCX runtimes :)

[setjmp / longjmp and Zero-Cost Exceptions —mp]

Abstract does not hide predefined operator

From: Stefan.Lucks <Stefan.Lucks@uni-weimar.de>
Date: Fri, 12 Nov 2010 16:04:27 +0100
Subject: Abstract operator does not hide predefined operator
Newsgroups: comp.lang.ada

Hi all,

I've implemented a small generic package for modular arithmetic. Note that Ada's "mod N" types define addition subtraction and multiplication right, but use the integer division for "/" instead of the proper modular division, where A/B is A * Inverse(B). I also tried to get rid of the unary "not" operator, whose purpose I don't understand for general modular arithmetic. (I understand that it is useful if the modulus is a power of two, just like the binary xor operator.)

This is the spec of my generic package:

```
generic
   type Mod_T is mod <>;
package Mod_Arith is
   type Modular is new Mod_T;
   function "/"(Left, Right: Modular)
      return Modular;
   function Inverse(Value: Modular)
      return Modular;
   function "not"(Value: Modular)
      return Modular is abstract;
end Mod_Arith;
```

For testing that package, I first instantiated it:

```
   type M3343 is mod 3343;
   package M is new
      Mod_Arith(Mod_T => M3343);
   use type M.Modular;
   S,T,U: M.Modular;
```

(Sidenote: 3343 is a prime, so addition and multiplication over M.Modular are field operations, mathematically.)

Now, my freshly defined division works as expected. The multiplication (and addition and subtraction) are inherited from the type M3343:

```
   S := 3;
   T := S/(2*S); -- T becomes the
                 -- multiplicative Inverse
                 -- of 2 mod 3343;
                 -- the same expression's
                 -- result in M3343 would
                 -- be zero.
   Ada.Text_IO.Put_Line(
      M.Modular'Image(T) &
      M.Modular'Image(M.Inverse(2)));
   -- The output is "1672 1672",
   -- as expected.
```

But, unfortunately, the compiler (GNAT) also accepts the following:

```
   U := not S; -- this should not be
               -- possible, because the
               -- "not" operator been
               -- defined abstract ... but
               -- instead, the predefined
               -- "not" from M3343 seems
               -- to be used
   Ada.Text_IO.Put_Line(
      M.Modular'Image(U));
   -- The output is "3339", which actually
   -- is not(M3343(S)) = 3342-S.
```

What can I do to get rid of the predefined "not"? If I change the "not" from an abstract function to a function always raising an exception, the

```
   U := not S;
```

raises the desired exception. But I would prefer to be told at compile time that I must not use "not", rather than at run time. Also, I would prefer not to define the type Mod_Arith.Modular as a private type.

Any ideas?

From: Adam Beneschan <adam@irvine.com>
Date: Fri, 12 Nov 2010 08:56:02 -0800 PST
Subject: Re: Abstract operator does not hide predefined operator
Newsgroups: comp.lang.ada

[…]

This appears to be a GNAT bug; the inherited "not" subprogram of Modular, which is inherited from the predefined function of M3343, exists but should be hidden from all visibility, and therefore cannot be a possible meaning of "not" in the expression "not S". So what you're doing should work, I think. (P.S. I've confirmed that the version of GNAT I'm using also accepts the program if I write out M instead of making it a generic instantiation, and if I use "use" instead of "use type". So neither of those appears to be an issue.)

From: Alex Mentis <asmentis@gmail.com>
Date: Fri, 12 Nov 2010 17:39:21 +0000 UTC
Subject: Re: Abstract operator does not hide predefined operator
Newsgroups: comp.lang.ada

[…]

Agree that it smells buggy. It seems that the problem only affects making unary operators abstract ("-" and "not"). Testing with "+", "-", "/", "*", "and", "or", and "xor" operators as abstract refused to compile as expected for me.

On discriminants of tagged types

From: Maciej Sobczak <maciej@msobczak.com>
Date: Wed, 27 Oct 2010 05:16:39 -0700 PDT
Subject: Discriminants of tagged types
Newsgroups: comp.lang.ada

Hi,

I'm not sure if I have already complained about it, but as I have hit that problem again, here it goes:

GNAT says that discriminants of tagged types cannot have default values.

So this is OK:

```
type T (A : Integer := 0) is null record;
```

but this is an error:

```
type T (A : Integer := 0) is
   tagged null record;
```

I find it disturbing and artificial, but perhaps I don't see the big picture. What is the rationale for this limitation?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 27 Oct 2010 05:34:26 -0700 PDT
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

[…]

If one discriminant has a default value, then all discriminants must have default values (ARM 3.7(9.1/2)). The reason is explained in the Annotated Ada Reference Manual.

http://www.adaic.com/standards/05aarm/html/AA-3-7.html

Tagged types have, by definition, a hidden discriminant which is the tag. The tag has no default value and cannot have one, for that would allow changing the type of an object at run time (through assignment of an aggregate). Therefore, *additional* discriminants cannot have default values either.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 27 Oct 2010 15:19:46 +0200
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

> The tag has no default value and cannot have one, for that would allow changing the type of an object at run time (through assignment of an aggregate).

No, it would not because you cannot assign a value of another type.

> Therefore, *additional* discriminants cannot have default values either.

I think there were concerns about implementation difficulties when the derived type adds new discriminants and/or defines some inherited ones.

Anyway the limitation is indeed annoying.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Wed, 27 Oct 2010 09:52:48 -0400
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

[…]

> No, it would not because you cannot assign a value of another type.

I think Ludovic means that the Tag could change. That is, tags are sort of like discriminants, so if Tags could have defaults, then we can guess that they would inherit the strange rule "can change if defaulted". So this would work:

```
A : T1;
B : T'Class := A;
C : T2;
B := C; -- Raises an exception.
```

[…]

> Anyway the limitation is indeed annoying.

Then you won't like my opinion, which is that this feature should never have existed even for untagged types. It's useful, but it's just not worth the trouble, in terms of programmer confusion, and implementation difficulty. Also non-portability:

```
type T(Length: Natural := 0) is
  record
    S: String(1..Length);
  end record;
```

This works on some implementations, and not others.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 27 Oct 2010 16:12:42 +0200
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

> […] So this would work:

But this is an unrelated case. The tag of a specific type cannot be changed regardless its view as a discriminant.

Considering the class-wide type, its discriminants are unknown, it is an open set. So the standard rules here do not apply anyway. BTW, if we considered class-wide assignment dispatchng, we would come to the case of a specific type when the tags are same, or to Constraint_Error when they differ.

And independently on defaults:

```
A : T1;
B : T2;
C : T'Class := A;
D : T'Class := B;
C := D;
```

> […] Then you won't like my opinion, which is that this feature should never have existed even for untagged types. […]

It might wonder you, but I wholeheartedly agree with you, but for another reason. The problem default values are supposed to solve IMO, should be solved by constructors and user-defined aggregates. If they existed and for any type, the programmer could provide variable list of arguments passed to the constructor.

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 27 Oct 2010 15:54:17 +0200
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

> […] GNAT says that discriminants of tagged types cannot have default values.

And it is right.

> I find it disturbing and artificial, but perhaps I don't see the big picture. What is the rationale for this limitation?

Default values of discriminants are not "just" default values: they define another form of record, the polymorphic record.

Tagged types define a different kind of polymorphism, through inheritance.

Having two orthogonal kinds of polymorphism in the same data structure would be difficult to manage, both from a design and an implementation point of view.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Wed, 27 Oct 2010 09:44:17 -0400
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

[…]

> GNAT says that discriminants of tagged types cannot have default values.

Well, Ada says that. GNAT is just following orders. ;-)

> but this is an error:

```
type T (A : Integer := 0) is tagged null record;
```

We tried very hard during Ada 9X to allow this, but we kept running into semantic difficulties, which required more and more complicated rules to fix. One day, Tucker said to me (or I said to Tucker -- I don't remember which), let's just outlaw this. We both agreed it was a big simplification.

Sorry, I don't remember in detail what the semantic difficulties were. Ludovic's explanation seems as good as any.

I believe Ada 2012 will allow defaults for LIMITED tagged types, and I think (not sure) it's already implemented in GNAT under -gnat2012 mode. The limited case is easy, because the weird rule that says "defaulted discriminants can change" isn't true for limited types, because assignment statements are forbidden.

From: Adam Beneschan <adam@irvine.com>
Date: Wed, 27 Oct 2010 08:06:24 -0700 PDT
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

[…]

AARM 3.7(9.d): "Defaults for discriminants of tagged types are disallowed so that every object of a tagged type is constrained, either by an explicit constraint, or by its initial discriminant values. This substantially simplifies the semantic rules and the implementation of inherited dispatching operations. For generic formal types, the restriction simplifies the type matching rules. If one simply wants a 'default' value for the discriminants, a constrained subtype can be declared for future use."

One issue with untagged types with default discriminants is that if you define a type like that:

```
type Rec (Discr : Integer := 0) is
  record…
```

and declare a procedure with an IN OUT parameter of that (unconstrained) type:

```
procedure Proc (R : in out Rec) is
begin
  …
  R := (Discr => R.Discr + 1,
        [other components => …])
   -- whether this is allowed depends on
   -- the actual parameter
end Proc;
```

you could declare both an unconstrained and constrained object of that type and pass them both to Proc:

```
R1 : Rec;
R2 : Rec(10);
Proc (R1);
Proc (R2);
```

Proc is allowed to assign a new value to R that changes the discriminant, but it has to know that if R2 is the actual parameter, the discriminant is not allowed to change. This requires additional information to be passed to Proc so that it knows whether to raise Constraint_Error when the assignment to R happens. Apparently it was felt that where tagged types, class-wide types, and dispatching were involved, this would make things too complicated.

This may seem "annoying" to those who think that a default discriminant is *just* a default (rather than creating a different kind of type that allows the creation of both constrained and unconstrained objects). But the last sentence of the AARM note shows that the workaround is simple enough. (Personally, I would have preferred that the two concepts not be mixed, and that there be two syntaxes---one for specifying a default value for discriminants, and another to specify that unconstrained objects of the type can be created. It's not the only case where having one syntax do "double duty" ends up causing problems. I think AI05-154 is a symptom of this, in which putting a constraint on a nominal array subtype not only specifies a constraint but also tells the compiler that an 'Access can't be used as an access-to-unconstrained-array value, causing headaches when you want to include a constraint but *also* want the ability to have an access-to-unconstrained-array point to the object; another case, perhaps, where "double duty" may have seemed like a clever solution to avoid extra syntax, but turned out not to be such a good idea… But I'm starting to rant a bit.)

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Wed, 27 Oct 2010 12:06:52 -0400
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

[…]

> In the mean time I have solved the problem by realizing that "taggedness" need not be public - in my case the type was tagged, because it was controlled (hate to repeat it, but the interaction between these two unrelated language features is really annoying).
>
> I don't have to expose the controlled nature of the type in the public view and moved the "tagged" keyword to private part. Interestingly, this allowed me to use default values for discriminants *and* have the controlled type.

That doesn't sound right. What does your example look like, and what compiler compiled it without error?

The latest GNAT says:

```
1. with Ada.Finalization; use
   Ada.Finalization;
2. package Eg is
3.    type T (X: Boolean := False) is
         private;
4. private
5.    type T (X: Boolean := False) is new
         Controlled with null record;
>>> discriminants of tagged type cannot
    have defaults
6. end Eg;
6 lines: 1 error
```

[…]

> (except that now the "distinguished receiver" notation is gone…)

I'm not a big fan of that notation. An awful lot of compiler work for a little bit of syntactic sugar (or maybe syntactic salt).

From: Maciej Sobczak <maciej@msobczak.com>
Date: Wed, 27 Oct 2010 14:05:32 -0700 PDT
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

> That doesn't sound right.

I was surprised, too.

>What does your example look like,

Much more complex, but reproducible with your code.

> and what compiler compiled it without error?

GPL 2009 (20090519)

> The latest GNAT says: […]

My compiler eats this stuff without even blinking. It complains only when I add the "tagged" keyword in line 3, between "is" and "private".

> […] I'm not a big fan of that notation. […]

Protected objects, tasks and records already supported it. It is not a new idea, so it wasn't that big deal for compiler writers, I guess. I treat it as a unification of syntax across these similar features - quite a valid goal in language design.

This is especially reasonable if you take into account that in Ada 2005 protected types can derive from interfaces. It would be very inconsistent not to allow the same calling syntax across the whole hierarchy, including the class-wide type.

```
type Base is interface;
procedure Do_Something (
    X : in out Base) is abstract;
protected type Derived is new Base
    with procedure Do_Something;
end Derived;
D : Derived;
B : Base'Class := D;
D.Do_Something;  -- this was always
                 OK, D is protected
B.Do_Something;  -- this *should* be
           OK as well for consistency
```

From: Simon Wright <simon@pushface.org>
Date: Wed, 27 Oct 2010 22:28:02 +0100
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

[…]

This compiles without complaint with GNAT GPL 2010 & GCC 4.5.0:

```
package Privately_Tagged is
   type T (X : Integer := 42) is private;
private
   type T (X : Integer := 42) is tagged
      null record;
end Privately_Tagged;
```

(and so does your example).

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Wed, 27 Oct 2010 20:35:44 -0400
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

[…]

OK, well then it's a compiler bug that has since been fixed.

Now that I think about it, I think I remember a recent bug in this area.

I'll bet if you tried hard enough, you'd find some cases where the compiler generated code that did strange things (before the bug was fixed, I mean).

> Protected objects, tasks and records already supported it.

Records never supported this notation for calls, which I thought was what we're talking about.

I wasn't a big fan of it for tasks or protected objects, either. But at least for these things, a "distinguished receiver" does make sense, because there's only one object getting locked. And because inside the operation, you can only see inside that one object. That is, in X.Y(Z), X is special in the protected object case, but X and Z are just normal parameters in the normal tagged case.

> … It is not a new idea, so it wasn't that big deal for compiler writers, I guess.

Well, it was. Maybe it wouldn't have been if this new syntax had been invented in the first place, but adding this sort of thing into an existing compiler can be quite a pain.

> I treat it as a unification of syntax across these similar features - quite a valid goal in language design. […]

Good point. But I'd prefer to unify the syntax in the other direction.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Thu, 28 Oct 2010 01:55:09 -0700 PDT
Subject: Re: Discriminants of tagged types
Newsgroups: comp.lang.ada

[…]

And I would accept that other direction without any criticism, really.

The problem is that time has only one direction and now there is no other option than to allow the dot syntax for everything.

In its current state, the language is inconsistent and only more difficult to learn.

Ada for Android

From: Riccardo Bernardini <framefritti@gmail.com>
Date: Sun, 7 Nov 2010 02:48:25 -0800 PST
Subject: Crazy idea: Ada for Android? (A4A?)
Newsgroups: comp.lang.ada

Dear all,

I just read an article (on a paper magazine, so no link available) about Android. As I understand, Android applications are executed by Dalvik, a specially-tailored Java VM and that there is a converter from Java VM to Dalvik. Since jgnat translates Ada to JVM… The idea of being able to write Android applications in Ada is too attractive… Do you believe that it is too ambitious a goal to try to bring Ada to the Android world? I have also a nice acronym with minimal clashes: A4A […]

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Sun, 07 Nov 2010 19:02:39 +0100
Subject: Re: Crazy idea: Ada for Android? (A4A?)
Newsgroups: comp.lang.ada

[…]

Yes. That »converter« (called aapt) is the default way to develop. […]

I tried Scala for Android and it did work. However: Scala needs a 6MB runtime library. aapt needs minute or so (on a 8 core MacPro) to convert that library and the result is 6 times big as an average Android application. So Scala is only an option for devices where you can pre-install the runtime library. That is: phones where you have root access.

So translated to Ada that means either “pragma No_Runtime;” (painful) or again pre-install the runtime-library (no good for customer devices).

Alternatively there is the NDK for native non-VM development. But that would only be do-able for Ada-Core itself.

From: Simon Clubley <clubley@eisner.decus.org-Earth.UFP>
Date: Mon, 8 Nov 2010 00:16:21 +0000 UTC
Subject: Re: Crazy idea: Ada for Android? (A4A?)
Newsgroups: comp.lang.ada

[…]

I'm on the verge of buying my first Android phone and I have considered this as well. However, I was thinking about the NDK level instead of the Java based SDK level.

> Alternatively there is the NDK for native non-VM development. But that would only be do-able for Ada-Core itself.

Why? OAR (the creators of RTEMS) have modified GCC so that Ada will run on RTEMS. The AVR-Ada people have modified gcc so that Ada will run on bare metal AVRs.

It's probably way too much work for me in light of other hobbyist projects I have pending, but I don't see why someone more familiar with the code base could not do it (at least in principle.)

On the precision of floating point types

From: J-P. Rosen <rosen@adalog.fr>
Date: Thu, 30 Sep 2010 08:58:52 +0200
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

> Is Ada's long_float like Fortran double precision? and short_float is like Fortran single precision (just called real in Fortran)?

There is nothing in the standard that guarantees this. Float and Long_Float are just floating point types, with greater accuracy for Long_Float.

On the other hand, package Interfaces.Fortran provides types Real and Double_Precision, which are required to match the corresponding Fortran types.

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Thu, 30 Sep 2010 01:46:56 -0700 PDT
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

> Thanks. Are you saying that in Ada I can't make double precision variables and single precision variables? (as in Fortran?)

[…]

Why don't you have a look in the RM, especially 3.5.7. There you can find the following definitions:

> {Float} In an implementation that supports floating point types with 6 or more digits of precision, the requested decimal precision for Float shall be at least 6.
>
> {Long_Float} If Long_Float is predefined for an implementation, then its requested decimal precision shall be at least 11.

You can define your own float types with the precision you need.

From: Mark Lorenzen <mark.lorenzen@gmail.com>
Date: Thu, 30 Sep 2010 02:59:42 -0700 PDT
Subject: Re: simple question on long_float/short_float

Newsgroups: comp.lang.ada

[…]

You should generally not use the predefined types for anything other than interfacing with other languages e.g. C or Fortran. Don't think in terms of "Ada has these types…" but define the types you need.

Ada is not Fortran, so don't code Fortran in Ada.

From: Peter C. Chapin <chapinp@acm.org>
Date: Thu, 30 Sep 2010 09:30:25 -0400
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

A common recommendation is to avoid using the built-in types directly.

Instead define your own:

```
type My_Floating_Type is digits 12;
```

If the implementation has a floating point type with at least 12 decimal digits of precision it will use it as the base type for My_Floating_Type and all will be well. If the implementation can't handle that much precision in any of its built-in types you will get a compile time error. For example you could try

```
type My_Floating_Type is digits 350;
```

But I doubt if there is any implementation that will honor that.

The types in Interfaces.Fortran are intended for the case where you are exchanging data with a corresponding Fortran compiler. It sounds like you might, in fact, be doing that.

From: J-P. Rosen <rosen@adalog.fr>
Date: Thu, 30 Sep 2010 15:45:16 +0200
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

> […] So, what is Ada's Float? 32 bit or 64 bits?

This is not specified by the definition of the language in Ada. Nor in Fortran, by the way.

With Ada, you can access all floating point types provided by the hardware, you are not constrained to single or double precision only (the VAX had 4 floating point formats, for example).

From: Jeffrey Carter <jrcarter@acm.org>
Date: Thu, 30 Sep 2010 08:37:52 -0700
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

In most languages (other than Ada), you have to use the numeric types provided by the language, and select the types that are the closest fit to the requirements of the problem. This is part of what is known as "translating the problem into the solution space", which makes the software harder to understand, since the reader has to understand both the problem and the translation chosen.

In Ada, one can declare numeric types to match the requirements of the problem.

This is part of what is known as "modeling the problem in the software", which is intended to make the software easier to understand, since there is no translation from problem to solution space to understand.

Ada's Float is defined as "at least 6 decimal digits of precision". This definition is imprecise, presumably to discourage its use in favor of user-defined types. If the precision of a floating-point type is important to the problem, you won't use a predefined type, but will declare your own with the desired precision.

Thinking in terms of declaring numeric types based on the needs of the problem, rather than choosing from language-supplied types, is part of the Ada Mind Set(tm).

From: Nasser M. Abbasi <nma@12000.org>
Date: Thu, 30 Sep 2010 11:22:29 -0700
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

This really opened my eyes to a very good point that I did not think of.

You are 100% right.

Very well said.

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 30 Sep 2010 08:56:47 -0700 PDT
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

> So, what is Ada's Float? 32 bit or 64 bits?

The language does not define this. The language is designed to be implemented on many processors, some of which have floats that are neither 32 nor 64 bits. Others have mentioned 80-bit floats. ADI SHARC has 40-bit floats. Some older systems (Honeywell 600 series e.g.) were based on 36-bit words and their floats were 36 bits. No idea what Crays define, but it wouldn't surprise me if there are processors designed for heavy scientific computation that would require floats of even greater precision than these.

Anyway, it was deliberate that the language did not define exactly what "Float", "Long_Float", and "Short_Float" are supposed to look like, so you shouldn't use those types if you care what kind of float you're getting. If you need the type to match a type in some other language implemented on that same processor, then use one of the types in Interfaces.Fortran or Interfaces.C or whatever. If you need your floating-point type to be specifically 32 or 64 or 57 bits for some other reason, then either define your own, or (if you don't care about portability) check to see what your particular compiler's definitions for that particular processor are. (Even on the same processor, the definition of "Float" may be different between different vendors' compilers, because the language doesn't define this.) You can use Float or Long_Float if you just want any floating-point type that meets or exceeds the minimum standard in 3.5.7(14-15) and don't really care how it's implemented. But only then.

From: Peter C. Chapin <chapinp@acm.org>
Date: Thu, 30 Sep 2010 17:21:02 -0400
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

I don't know about old Fortran but in Fortran 90 and above you can specify the characteristics of your floating point type and ask the compiler to find an underlying type that works for you.

Each compiler provides various "kinds" of floating point types.

Depending on the underlying machine there might be several. These kinds are identified by compiler specific integer constants (kind 1, kind 2, etc). Of course referring to them that way isn't portable... my floating point type that is labeled as "kind 1" might be nothing like the floating point type you are labeling as "kind 1." To get around this you can use the intrinsic function SELECTED_REAL_KIND. It looks like this:

```
INTEGER, PARAMETER ::
my_floating_kind =
SELECTED_REAL_KIND(6, 30)
```

This asks the compiler to find a floating point type with at least 6 digits of precision and a range that covers 10**30. The resulting kind (some number like 1, 2, 3, etc) is then given the name "my_floating_kind."

I declare variables like this

```
REAL(KIND = my_floating_kind) ::
  x, y, z
```

The kind value is like a type parameter. Thus REAL(KIND = 1) means you want the first kind of floating point type, REAL(KIND = 2) means you want the second kind. In the declaration above "my_floating_kind" is used meaning that you want whatever kind is necessary to get the desired precision and range.

Maybe that's kind 2 on my compiler and kind 17 on yours.

Fortran compilers are required (I think) to provide a double precision type but internally that type is just a particular kind (maybe kind 2?). In modern Fortran you never have to say "double precision" the term exists for compatibility and convenience.

We now return to our regularly scheduled discussion of Ada. :)

From: Peter C. Chapin <chapinp@acm.org>
Date: Fri, 01 Oct 2010 07:13:14 -0400
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

> This trend to update computer languages every few years seems to result in a language first starting as simple, with the basic features, then it gets more and more complicated and bloated as more features are stuffed in it to make it more cool and modern. With time computer languages become too complex to understand by mere mortals, unless one has a PhD in computer science.

I agree there definitely is that trend. Of course part of the problem is that old features can only be removed with extreme difficulty since most language communities want backward compatibility. Thus the "size" of a language tends to be a monotonically increasing function of time.

In fact, this is one reason why I appreciate Ada. Although the language is large and complicated, my perception is that new features are added to it conservatively and only after extensive deliberation. I'm not saying other communities don't deliberate extensively as well, but it seems like some languages (C++?) grow more quickly than is good for them.

Also Ada has a mechanism (pragma Restrictions) where one can selective remove features from the language depending on one's needs. I don't know of any other language that offers a standard way to make it smaller.

Finally there is SPARK. The language itself is actually quite simple and basic. Of course fully using the annotations and doing all the associated proofs introduces its own complexity, but much of that could be skipped if desired.

On float precision and IEEE 754

From: Nasser M. Abbasi <nma@12000.org>
Date: Sat, 02 Oct 2010 02:11:10 -0700
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

It says here

http://www.pegasoft.ca/resources/boblap/17.html

"GNAT provides interfacing packages for languages besides C.

Interfaces.Fortran contains types and subprograms to link Fortran language programs to your Ada programs. The GCC Fortran 77 compiler is g77.

As with GCC, most of the Fortran data types correspond identically with an Ada type. A Fortran real variable, for example, is the same as an Ada float, and a double precision variable is an Ada long_float. Other Ada compilers may not do this: if portability is an issue, always use the types of Interfaces.Fortran."

I wrote a small Ada program to print a value of a floating number which has the type Double_Precision, and printed the value to the screen.

I did the same in Fortran, same number, and printed the value to screen.

In Fortran, the value printed is that of IEEE754, and in Ada it also printed the same as Fortran. So I am happy to see that. Here is the code and the output:

---- Ada v1 ----

```
with ada.text_io;
with Interfaces.Fortran;
use Interfaces.Fortran;
procedure test is
   package my_float_IO is new
     Ada.Text_IO.float_IO(
       Double_Precision);
   use  my_float_IO;
  a : Double_Precision  := 2.3;
begin
  Put( item=>a,Fore=>25,Aft=>16) ;
end test;
```

---- fortran ----

```
program main
    implicit none
    double precision :: c=2.3D0
    write (*,'(F25.16)') c
end program main
```

---- output of the above in same order ---

```
2.2999999999999998E+00  ---- Ada
2.2999999999999998      ---- Fortran
```

Then I changed the Ada code to the following:

---- Ada v2 -----

```
with ada.text_io;
procedure test is
   type my_float_type is digits 16;
   package my_float_IO is new
     Ada.Text_IO.float_IO(
       my_float_type);
   use  my_float_IO;
  a : my_float_type := 2.3;
begin
  Put( item=>a,Fore=>25,Aft=>16) ;
end test;
```

and now the output is

```
2.3000000000000000E+00
```

1) So, it seems to me that Ada did use Fortran double in v1. since output is different than in v2. Unless I made a mistake in v2. Hence, I do not understand why the webpage above said that GNAT would use the same types. Can I change v2 to make it output the same as Fortran? I assume not, since it is not double to start with.

2) The reason I wanted to use Fortran double type, so I can compare the output of my Ada program to that of Fortran and some output in the textbook.

3) I need to read more about Ada Float type. Standard is IEEE754, so if Ada float does not use this, would this not make the analysis of floating point computation in Ada harder? Most textbooks and numerical stuff, assume IEEE754 computation done on floating points.

I also need to study more about Ada floats and how they work. What I do not understand yet, does Ada use IEEE 754 for its floats? Assuming it does, then it must do something additional at run-time to obtain the result it needs when one uses DIGITS nnn in the type definition for Ada float.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 2 Oct 2010 11:48:47 +0200
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

[…]

Hmm, try this:

```
with Ada.Text_IO; use Ada.Text_IO;
with Interfaces.Fortran;
use Interfaces.Fortran;
procedure Test is
   type My_Float_Type   is digits 16;
   type Long_Float_Type is digits 18;
   A : My_Float_Type    := 2.3;
   B : Double_Precision := 2.3;
   C : Long_Float_Type  := 2.3;
begin
   Put_Line ("A'Size=" &
         Integer'Image (A'Size) & " A=" &
         My_Float_Type'Image (A));
   Put_Line ("B'Size=" &
         Integer'Image (B'Size) & " B=" &
         Double_Precision'Image (B));
```

```
   Put_Line ("C'Size=" &
         Integer'Image (C'Size) & " C=" &
         Long_Float_Type'Image (C));
end Test;
```

On my Intel machine Double_Precision is 96 bits.

Then note that the output may differ not because the Ada type uses another machine type. Even if they are same, since you have specified 16 *decimal* digits, the output rounds the underlying binary representation to 16 decimal digits.

> 3) I need to read more about Ada Float type. Standard is IEEE 754, so if Ada float does not use this,

That depends on the machine. Float and Long_Float standard types are most likely IEEE 754.

> would this not make the analysis of floating point computation in Ada harder?

No, it makes it easier. It is advisable not to use IEEE 754 semantics, even if the machine is IEEE 754. E.g. if you declare your type as

```
type My_Float is new Float; -- Chances
      -- are high to get IEEE 754 here
```

To kill IEEE 754 semantics do it this way:

```
type My_Float is new Float
   range Float'Range; -- Only numbers!
```

> Most textbooks and numerical stuff, assume IEEE754 computation done on floating points.

I doubt it much. IEEE 754 is a normal floating-point + some non-numbers like NaN. As the name suggests, numeric computations are performed on *numbers*. IEEE 754 non-numbers might be useful for the languages with no means to deal with numeric errors. But in Ada you don't need that because Ada has numeric exceptions.

From the software design point of view, it is a very bad idea of IEEE 754 to postpone error handling till the end of computations or even forever.

You might compute something awfully long and complex just to get NaN in the end, not knowing what (and where) was wrong. In a closed loop system you might deliver NaN on actuators, what would they do?

Ada's numeric model follows the fundamental software design principle: you shall detect errors *early*.

> I also need to study more about Ada floats and how they work. What I do not understand yet, does Ada use IEEE 754 for its floats?

It likely uses IEEE 754 if the machine is IEEE 754, which is almost always.

From: Simon Wright <simon@pushface.org>
Date: Sat, 02 Oct 2010 21:09:26 +0100
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

> But in Ada you don't need that because Ada has numeric exceptions.

But GNAT doesn't raise numeric exceptions for floating-point computations resulting in Inf or NaN ('Machine_Overflows is False).

You can use 'Valid, or you can define the range as Dmitry noted above (range Float'Range, I think) in which case you get Constraint_Error on overflow.

From: Jeffrey Carter <jrcarter@acm.org>
Date: Sat, 02 Oct 2010 09:52:43 -0700
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

> […] Can I change v2 to make it output the same as Fortran? […]

Sure. Try

```
type My_Float is digits Interfaces.
   Fortran.Double_Precision'digits;
```

or

```
type My_Float is digits
   Long_Float'digits;
```

or even

```
type My_Float is digits 15;
```

all of which give My_Float the same "digits 15" declaration as Double_Precision and Long_Float.

> I also need to study more about Ada floats and how they work. What I do not understand yet, does Ada does use IEEE754 for its floats? […]

Ada compilers typically use an underlying hardware floating-point type to implement floating-point type declarations. That is IEEE-754 on many platforms, but not all.

From: Georg Bauhaus <rm-host.bauhaus@maps.futureapps.de>
Date: Sat, 02 Oct 2010 22:01:53 +0200
Subject: Re: simple question on long_float/short_float
Newsgroups: comp.lang.ada

In addition to that, compilers may generate different code for what only appears to be the "same platform": Using GNAT, if you add options -mfpmath=sse or -mfpmath=387, the compiler will issue instructions for a corresponding set of registers, respectively.

(What exactly this means w.r.t. IEEE-754 I don't know).

On the renaming of subprograms

From: Gene Ressler <gene.ressler@gmail.com>
Date: Sat, 25 Sep 2010 17:43:17 -0700 PDT
Subject: Renaming of procedures in a generic instantiation
Newsgroups: comp.lang.ada

I'm confused about an aspect of renaming. It boils out to this little example:

```
with Ada.Containers.Ordered_Sets;
package Foo is
  type Queue is private;
  procedure Add(Q : in out Queue;
                Item : in Integer);
  function Is_Empty(Q : Queue)
    return Boolean;
private
  package Queues is
    new Ada.Containers.
      Ordered_Sets(Integer, "<", "=");
  type Queue is new Queues.Set
    with null record;
end Foo;
package body Foo is
   procedure Add(Q : in out Queue;
                 Item : in Integer)
     renames Insert;
   function Is_Empty(Q : Queue)
     return Boolean
     renames Queues.Is_Empty;
end Foo;
```

The renaming in Add "finds" the correct procedure Insert, but the renaming of Is_Empty fails:

```
gnatmake foo.adb
gcc -c foo.adb
foo.adb:6:04: no visible subprogram
matches the specification for
"Is_Empty"
foo.adb:6:13: expected private type
"Ada.Containers.Ordered_Sets.Set"
from instance at foo.ads:14
foo.adb:6:13: found type "Queue"
defined at foo.ads:16
gnatmake: "foo.adb" compilation error
```

I guess I can see this error because a parameter type conversion is implied in the renaming. But then why does the renaming of Insert work correctly?

Running Win7 64-bit with:

gcc (GCC) 4.3.4 20090511 for GNAT GPL 2009 (20090511)

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Sun, 26 Sep 2010 09:54:15 +0300
Subject: Re: Renaming of procedures in a generic instantiation
Newsgroups: comp.lang.ada

[…]

The type derivation

```
type Queue is new Queues.Set
  with null record;
```

makes type Queue inherit the primitive operations of Queues.Set by implicitly declaring (within package Foo) corresponding operations on type Queue. These include the operation Insert with the profile:

```
procedure Insert (
  Container : in out Queue;
  New_Item  : in     Integer);
```

This profile matches that of Foo.Add, and so Foo.Add can be a renaming of this Insert. By the way, this implicitly declared Insert (on Queue) has the qualified name Foo.Insert, not Queues.Insert.

The type derivation also "tries" to declare implicitly the Is_Empty operation from Queues, with the profile:

```
function Is_Empty (Container : Queue)
  return Boolean;
```

but this operation clashes with (has the same name and same profile as) the operation Is_Empty that was already and explicitly declared in Foo (before the "private"), so they cannot both be visible. If I understand correctly, the rule RM 8.3(10/1) says that the first, explicit declaration overrides the second, implicit declaration (because the first declaration is not "overridable").

So what saves you in the Insert/Add case is that the operation names are different, which means there is no clash. It is a bit irritating that the "simpler" case, with the same names, does not work in the same way.

I have at times had the same problem, where I have been disappointed that implementing a private type as a derived type does not implicitly complete the declarations of matching operations, such as Is_Empty in this example. I would have liked to be able to write something like:

```
type Queue is new Queues.Set
  with null record
    overriding Is_Empty (Q : Queue)
      return Boolean;  -- Not Ada!
```

No doubt you have already found a solution for Is_Empty, but anyway here is what I would do:

```
function Is_Empty (Q : Queue)
  return Boolean
is
begin
  return Queues.Is_Empty (
    Queues.Set (Q));
end Is_Empty;
```

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sun, 26 Sep 2010 00:40:48 -0700
Subject: Re: Renaming of procedures in a generic instantiation
Newsgroups: comp.lang.ada

[…]

Or change the name of the operation (Empty comes to mind). I would probably use a wrapper record:

```
type Queue_Handle is
   Value : Queues.Set;
end record;
procedure Put (
     Onto : in out Queue_Handle;
     Item : in Integer) is
   -- null;
begin -- Put
   Onto.Insert (New_Item => Item);
end Put;
function Is_Empty (
      Queue : in Queue_Handle) is
   -- null;
begin -- Is_Empty
   return Queue.Value.Is_Empty;
end Is_Empty;
```

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sun, 26 Sep 2010 10:07:24 -0700
Subject: Re: Renaming of procedures in a generic instantiation
Newsgroups: comp.lang.ada

>>[…]

>> Onto.Insert (New_Item => Item);

>

> A nit: You surely meant

>

> Onto.Value.Insert (New_Item => Item);

Yes, of course. Luckily, the compiler catches these for you.

From: Gene Ressler <gene.ressler@gmail.com>
Date: Sun, 26 Sep 2010 07:52:58 -0700 PDT
Subject: Re: Renaming of procedures in a generic instantiation
Newsgroups: comp.lang.ada

[…]

Thanks for the clear and quick explanation. It makes sense, but, as you say, is a bit unsatisfying. I've run into several cases where I want to implement an abstraction with minor reshaping and restriction of an Ada.Containers instance. (The current one is event queues for a simulator.) As you've shown above, there are some quirks in this approach.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 26 Sep 2010 17:04:49 +0200
Subject: Re: Renaming of procedures in a generic instantiation
Newsgroups: comp.lang.ada

[…]

Delegation (in order to implement interfaces) is poorly supported in Ada. In real projects we have huge problems with this. About 40% of the code are wrappers, nasty, error prone, incredibly difficult to maintain.

From: Adam Beneschan <adam@irvine.com>
Date: Mon, 27 Sep 2010 12:23:00 -0700 PDT
Subject: Re: Renaming of procedures in a generic instantiation
Newsgroups: comp.lang.ada

> […]

> But then why does the renaming of Insert work correctly?

Because you didn't refer to the "Insert" defined in Queues, the way you did with Is_Empty. If you had declared this, it would be illegal in the same way:

```
procedure Add(Q : in out Queue;
              Item : in Integer)
  renames Queues.Insert;
```

Obviously, you can't solve the problem for Is_Empty by removing the "Queues." prefix:

```
function Is_Empty(Q : in out Queue)
   return Boolean
  renames Is_Empty;
```

I found that this compiles, but it's obnoxious:

```
with Ada.Containers.Ordered_Sets;
package Foo is
  type Queue is private;
  procedure Add(Q : in out Queue;
                Item : in Integer);
  function Is_Empty(Q : Queue)
    return Boolean;
private
  package Queues is
    new Ada.Containers.
      Ordered_Sets(Integer, "<", "=");
  package Dummy_Package is
    type Dummy_Queue is new
      Queues.Set with null record;
    function Rename_Empty(
      Q : Dummy_Queue)
      return Boolean
    renames Is_Empty;
  end Dummy_Package;
  type Queue is new Dummy_Package.
    Dummy_Queue with null record;
end Foo;
package body Foo is
  procedure Add(Q : in out Queue;
                Item : in Integer)
    renames Insert;
  function Is_Empty(Q : Queue)
      return Boolean
    renames Rename_Empty;
end Foo;
```

Just writing Is_Empty to call Queues.Is_Empty, as Niklas suggested, seems better than this. I'd also add "pragma Inline(Is_Empty)" to the private part of Foo to possibly prevent extraneous call code from being generated.

I had thought that someone (possibly me) once proposed adding the ability to declare that a subprogram renames the hidden subprogram that it overrides, e.g.

```
function Is_Empty(Q : Queue)
   return Boolean renames <>;
```

but I can't find anything like that in my mail records. Randy did once mention using something like Is_Empty'Parent as a stand-in for the call to the parent routine; he wasn't thinking of a renaming context, but it might work here:

```
function Is_Empty(Q : Queue)
     return Boolean
   renames Is_Empty'Parent;
```

In any event, however, it seems to be a minor enough problem with a simple enough workaround that it's not worthwhile to change the language.

On libflorist and ioctl

From: Francesco Piraneo Giuliano <fpiraneo@gmail.com>
Date: Thu, 21 Oct 2010 02:13:23 -0700 PDT
Subject: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

In Linux we have the ioctl function that allows to control or get informations about an open device.

Questions:

- I don't understand if the ioctl is a function of POSIX compliant OS'es so it can be handled by Ada libflorist or is a Linux-specific function that we have to import in Ada with a pragma Import;
- In the case that ioctl is a standard POSIX function, I'm very glad if someone can address me to the right function in libflorist… I've not found anything similar.

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 21 Oct 2010 03:20:23 -0700 PDT
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

The title of your post is quite correct; ioctl is indeed black magic :)

POSIX[1] defines this function, as "obsolescent" mind you, along with several values of the request parameter but most operating systems add nonstandard values to the standard ones. Most of these values are appropriate only for certain kinds of device. Depending on the value of the request parameter, ioctl performs entirely different things and takes a different arg parameter; it is a variadic function which has no equivalent in Ada. So, calls to ioctl are sometimes portable (if request has one of the standard values), sometimes non-portable across operating systems, sometimes non-portable across devices on a single machine, and always obsolescent!

Therefore, a thick Ada binding would not expose ioctl directly but instead offer a different subprogram (with the appropriate parameters) for each supported value of the request parameter.

This may explain why Florist has no binding to ioctl. GNAT.Sockets has one in the body off the package but it is used for only one specific value of the request parameter. So I suggest you create a binding only for the specific purpose you need ioctl for and use that.

[1] http://www.opengroup.org/onlinepubs/9699919799/

[…]

See [1] above for the official documentation of POSIX. The additional, non-standard values of "request" along with the parameter profiles are in the ioctl_list(2) man page installed on your machine. See that and despair…

From: Francesco Piraneo Giuliano <fpiraneo@gmail.com>
Date: Thu, 21 Oct 2010 04:31:53 -0700 PDT
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

But my application […] has to run under Linux so to collect some data about Linux framebuffer I have to use ioctl; the only solution is to write all low level interfacing (open the device, get informations about, map it into memory) in C then write the upper level in Ada?

If this is true I have some philosophical concerns about… ;-)

[…]

From: Mark Lorenzen <mark.lorenzen@gmail.com>
Date: Thu, 21 Oct 2010 04:50:37 -0700 PDT
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

No you don't need to write anything in C. […]

It's difficult to know your exact requirements since ioctl is a variadic system call. Maybe you can post the equivalent C-code showing how you call ioctl in C and then we can translate it into Ada.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 21 Oct 2010 05:04:47 -0700 PDT
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

> […]

> the only solution is to write all low level interfacing (open the device, get informations about, map it into memory) in C then write the upper level in Ada?

No; I wrote:

>> […] So I suggest you create a binding only for the specific purpose you need ioctl for and use that.

What I meant by this is:

```
function Fixed_Info (
     Framebuffer : in
     POSIX.IO.File_Descriptor)
  return Fixed_Framebuffer_Info
is
   function ioctl
    (Framebuffer : in
        POSIX.IO.File_Descriptor;
     Request     : in Interfaces.C.int;
     Result      : access
        Fixed_Framebuffer_Info)
    return Interfaces.C.int; -- local,
                specific-purpose binding
   pragma Import (C, ioctl, "ioctl");
   Result : aliased
        Fixed_Framebuffer_Info;
begin
   if ioctl (Framebuffer,
             FBIOGET_FSCREENINFO,
             Result'Access) /= 0
   then
     raise Ioctl_Error;
   end if;
   return Result;
end Fixed_Info;
```

[…]

From: Michael Bode <m.g.bode@web.de>
Date: Fri, 22 Oct 2010 22:16:00 +0200
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

Look here http://www.pegasoft.ca/resources/boblap/book.html for information on how to program for Linux in Ada. You can call ioctl from an Ada program, you only have to arrange the parameters. […]

From: Florian Weimer <fw@deneb.enyo.de>
Date: Sat, 23 Oct 2010 22:26:51 +0200
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

ioctl is a varargs function. You cannot call C varargs functions directly from Ada.

You need to write a small wrapper with a fixed number of arguments.

In addition, the ioctl constants are often difficult to extract from the header files. The best way to get them seems to be a small C program which just prints them.

So writing some C code is unavoidable here.

From: Simon Wright <simon@pushface.org>
Date: Sun, 24 Oct 2010 12:08:28 +0100
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

It could print them as an Ada spec; no manual handling required!

From: Frank J. Lhota <FrankLho@rcn.com>
Date: Sun, 24 Oct 2010 08:41:14 -0400
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

> […] So writing some C code is unavoidable here.

Actually, you can call a varargs function such as ioctl from Ada.

Declare each ioctl profile that you need, e.g.

```
-- ioctl for commands with no
-- additional arguments
function Ioctl (Fd  : in Interfaces.C.Int;
                Cmd : in Interfaces.C.Int)
         return Interfaces.C.Int;
   pragma Import (C, Ioctl, "ioctl");
-- ioctl for commands with an
-- additional argument
-- for returning the device status
function Ioctl (Fd    : in Interfaces.C.Int;
                Cmd : in Interfaces.C.Int;
                Status : access
                         Interfaces.C.Int)
         return Interfaces.C.Int;
   pragma Import (C, Ioctl, "ioctl");
```

From: Florian Weimer <fw@deneb.enyo.de>
Date: Sun, 24 Oct 2010 19:56:59 +0200
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

Perhaps it seems to work for you, but this is not portable.

There are popular targets where the varargs calling convention is markedly different from the non-varargs calling convention, such as amd64.

It might still work by accident, but all bets are off, really.

From: Florian Weimer <fw@deneb.enyo.de>
Date: Sun, 24 Oct 2010 19:58:04 +0200
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

> […] It could print them as an Ada spec; no manual handling required!

Sure (and I did that in similar cases).

By the way, is there a good way to integrate such source code generation into the gnatmake project-based build system?

From: Simon Wright <simon@pushface.org>
Date: Sun, 24 Oct 2010 19:36:34 +0100
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

I hadn't realised this: but a quick scan of the GNAT sources confirms:

in socket.c,

```
* Wrapper for ioctl(2), which is a
variadic function */
int __gnat_socket_ioctl (int fd, int req,
                         int *arg) {
#if defined (_WIN32)
  return ioctlsocket (fd, req, arg);
#else
  return ioctl (fd, req, arg);
#endif
}
```

From: Frank J. Lhota <FrankLho@rcn.com>
Date: Sun, 24 Oct 2010 20:45:26 -0400
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

This is because of a Microsoft portability problem; in Winsock (the windows socket facility), the function for performing ioctl on sockets is named ioctlsocket instead of ioctl. This is one of many areas where Winsock differs just enough from the standards to require heavy use of "#if" preprocessing directives to maintain portability. This is why Winsock is by far my least favorite part of the Win32 API. At any rate, this problem is not an Ada issue.

From: Frank J. Lhota <FrankLho@rcn.com>
Date: Sun, 24 Oct 2010 21:13:58 -0400
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

The Microsoft documentation seems to say that varargs is _not_ markedly different; see

http://msdn.microsoft.com/en-us/library/dd2wa36c%28v=VS.80%29.aspx

I will concede, however, that these Import pragmas may not be portable to all platforms, but that is true of Import pragmas in general. After all, the calling convention for ioctl can vary from platform to platform, not to mention which calling conventions are supported by the Ada compiler.

I am absolutely sure that my pragmas will not work with MS Windows, for an more elementary reason: Win32 / Win64 does not support the ioctl function at all! Instead, the function DeviceIoControl is used to query configure most devices, and the function ioctlsocket is used to query / configure sockets. See

http://msdn.microsoft.com/en-us/library/aa363216.aspx

http://msdn.microsoft.com/en-us/library/ms738573%28VS.85%29.aspx

From: Florian Weimer <fw@deneb.enyo.de>
Date: Mon, 25 Oct 2010 20:56:03 +0200
Subject: Re: The "black magic" of ioctl
Newsgroups: comp.lang.ada

[…]

There are additional requirements for passing floating-point arguments.

It's different from Linux that this will only bite you when you actually pass floating-point arguments which are used in the callee in a certain way (which should happen with ioctl).

On -fstack-check in GNAT/GCC

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Mon, 25 Oct 2010 22:15:21 +0200
Subject: GNAT/GCC and -fstack-check
Newsgroups: comp.lang.ada

[…]

I would like to know if someone had the same I get.

First a few words about the context. I wanted to get rid of having two compilers, GNAT from AdaCore and GNAT from MinGW, to keep only one, which I choose to be MinGW for multiple reasons. When I wanted to build some applications which was built before with GNAT from AdaCore with now the GNAT from MinGW, I get a surprising behavior.

I was using -fstack-check in the GPR file. At compile time, I get these kind of messages:

```
xxxxx.adb: In function 'XXX.YYY'
xxxxx.adb:1:22: warning: frame size
  too large for reliable stack checking
xxxxx.adb:1:22: warning: try reducing
  the number of local variables
```

Where XXX.YYY is actually not a function but a package (do not know why the message says it is a function). Moreover, there was not a lot of local variables… just a few.

None of the compiled applications could launch and were all crashing right at start up.

After some time (I checked that the package raising these warnings was the

ones whose initialization was crashing) I decided to remove the -fstack-check option from the GPR file. There were then no more warnings and compiled applications could run properly.

I wonder if this is GCC related or GNAT related. The MinGW is the last one (decided to update it in the while), it seems to use GCC 4.5 and GNAT 2009.

I suppose it is based on GNAT 2009, as "gnatmake -v" does not know about the "-gnat12" option.

Did anybody experienced something similar with GNAT and -fstack-check? Or are there some reasons to suppose this is more GCC related than GNAT related?

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Tue, 26 Oct 2010 07:11:51 +0200
Subject: Re: GNAT/GCC and -fstack-check
Newsgroups: comp.lang.ada

Three years ago, Niklas Holsti seems to have raised the same question:

http://www.rhinocerus.net/forum/lang-ada/116262-gnat-fstack-check-does-work.html

He said:

[In "GNAT and -fstack-check, does it work?"]

I have been using those options for a while, but I was recently hit by a bug in which stack overflow made the program abort with Segmentation Violation (signal 11 = SIGSEGV) instead of the expected Storage_Error exception.

Which is exactly what I get too. I get a signal, not an exception, otherwise I wouldn't get a Windows crash-box.

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Tue, 26 Oct 2010 07:23:57 +0200
Subject: Re: GNAT/GCC and -fstack-check
Newsgroups: comp.lang.ada

[…]

Seven years ago, there was a bug report for the same: http://gcc.gnu.org/bugzilla/show_bug.cgi?id=10127

(not the same Niklas was talking about, this one is even older).

It was marked as solved… I feel it was either not really or else it is back.

By the way, I am surprised the Ada environment does not catch this signal to turn it into an exception. I would understand if this was a compiler bug, but what I now see as a compiler bug, is that the program does not catch the signal and let the program crash.

I will either investigate more to understand what is going wrong with the stack with that package, or else will simply drop the use of -fstack-check (as I am not sure this make the program crash because there is indeed a real stack overflow somewhere).

Any opinions and thoughts on that topic appreciated.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 26 Oct 2010 10:04:53 +0200
Subject: Re: GNAT/GCC and -fstack-check
Newsgroups: comp.lang.ada

[…]

This GCC back-end problem is old and has been reported multiple times with several variants:

http://gcc.gnu.org/PR10127

http://gcc.gnu.org/PR13182 (still open)

http://gcc.gnu.org/PR13757 (duplicate of PR10127)

http://gcc.gnu.org/PR20548

http://gcc.gnu.org/PR39306 (duplicate of PR10127)

I first learned of this problem thanks to PR20548 and consequently removed -fstack-check from all Ada packages in Debian.

PR10127 and PR20548 were both fixed in November 2009 in the trunk, therefore this fix should be in GCC >= 4.5.0 (per http://gcc.gnu.org/develop.html#timeline).

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Tue, 26 Oct 2010 10:16:09 +0200
Subject: Re: GNAT/GCC and -fstack-check
Newsgroups: comp.lang.ada

[…]

But the GCC I now use is :

gcc.exe (GCC) 4.5.0

Copyright (C) 2010 Free Software Foundation, Inc.

From: Ludovic Brenta <ludovic@ludovic-brenta.org
Date: Tue, 26 Oct 2010 08:11:17 -0700 PDT
Subject: Re: GNAT/GCC and -fstack-check
Newsgroups: comp.lang.ada

[…]

Ah, there is also

http://gcc.gnu.org/PR43013

Read the comments from Eric Botcazou…

[excerpts of the mentioned comments follow —mp]

I'll write a book about -fstack-check someday…

-fstack-check was severely broken during the GCC3 -> GCC4 transition and, despite years of patches posting and pinging, only GCC 4.5 has the beginning of a working implementation, so anything between 4.0 and 4.4 must be forgotten since totally broken.

And, again despite posted patches, the 4.5 implementation only restores the old implementation available in the 3.x series which doesn't work for this case: […]

Granted, we now generate wrong code instead of erroring out, but it's again a fallout of the GCC3 -> GCC4 breakage, not of my patches.

This will be fixed once the improved implementation is merged.

ETA is 4.6, for x86/x86-64 at least.

On array initialization in SPARK

From: Peter C. Chapin <chapinp@acm.org>
Date: Thu, 28 Oct 2010 03:13:02 -0700 PDT
Subject: Array initialization in SPARK
Newsgroups: comp.lang.ada

Hello!

Consider the following type definition:

```
type Matrix is array (Natural range <>,
                      Natural range <>) of
                      Types.Float8;
```

Now consider the following "obvious" implementation of a matrix transpose procedure:

```
procedure Matrix_Transpose(
            In_Matrix : in Matrix;
            Out_Matrix : out Matrix) is
  begin
    for I in Natural range
             Out_Matrix'Range(1) loop
      for J in Natural range
             Out_Matrix'Range(2) loop
        Out_Matrix(I, J) :=
             In_Matrix(J, I);
      end loop;
    end loop;
  end Matrix_Transpose;
```

This procedure makes assumptions about the relative sizes of the two matrix objects. I intend to assert those assumptions using preconditions. However, my question right now is about the initialization of Out_Matrix. As written the Examiner complains that the "initial undefined value of Out_Matrix is used in the definition of Out_Matrix" (or words to that effect).

I understand the issue. The assignment inside the loop only assigns to a single matrix element at a time. Thus, for example, the first time it executes the resulting value of Out_Matrix has only one defined element; the rest of the matrix has its "initial undefined value." I understand that SPARK works this way because it can't be expected to track individual array elements because array indexes are dynamic constructs.

In the past when I've had this problem I've just used an aggregate to initialize the array. In full Ada I can do

```
Out_Matrix :=
   (others => (others => 0.0));
```

However, SPARK isn't happy with this. I'm having trouble figuring out what would make SPARK happy here. Actually it would be even better if I could convince the Examiner that the overall effect of the two loops is to initialize the entire Out_Matrix. I'm not keen about spending execution time with an aggregate initialization only to overwrite the initial values anyway. In fact, SPARK complains about doing that sort of thing with scalar values so it certainly doesn't seem like the "SPARK way."

From: Phil Thornley <phil.jpthornley@gmail.com>
Date: Thu, 28 Oct 2010 05:47:53 -0700 PDT
Subject: Re: Array initialization in SPARK
Newsgroups: comp.lang.ada

> […]

> However, SPARK isn't happy with this. I'm having trouble figuring out what would make SPARK happy here.

I don't think that you can get SPARK to accept an aggregate assignment for an unconstrained array. Section 4.1 of the LRM says:

"SPARK excludes most whole-array operations on unconstrained array objects, in order that rules relating to index bounds may be statically checked."

> […] In fact, SPARK complains about doing that sort of thing with scalar values so it certainly doesn't seem like the "SPARK way."

It's far better to use the accept annotation (which is there for this sort of situation):

```
begin
  for I in Natural range
           Out_Matrix'Range(1) loop
    for J in Natural range
           Out_Matrix'Range(2) loop
    --# accept F, 23, Out_Matrix, "All
      of array is assigned in the loop.";
       Out_Matrix(I, J) :=
          In_Matrix(J, I);
    --# end accept;
    end loop;
  end loop;
  --# accept F, 602, Out_Matrix,
   Out_Matrix, "All of array is assigned
   in the loop.";
  end Matrix_Transpose;
```

[…]

From: Peter C. Chapin <chapinp@acm.org>
Date: Thu, 28 Oct 2010 07:51:15 -0700 PDT
Subject: Re: Array initialization in SPARK
Newsgroups: comp.lang.ada

> It's far better to use the accept annotation (which is there for this sort of situation):

Thanks for the suggestion. I can see the value of accept here.

In this case it's pretty obvious that the entire output matrix is being assigned. I have a couple of other procedures with a similar issue but where it is much less obvious that they are setting values for all matrix elements (they are supposed to be doing so). I can see that while accept is a useful tool one would have to be careful about sprinkling them around too liberally. I'll take a closer look at those other procedures and see if a little code reorganization might be desirable.

Thanks again!

From: Peter C. Chapin <chapinp@acm.org>
Date: Thu, 28 Oct 2010 08:23:19 -0700 PDT
Subject: Re: Array initialization in SPARK
Newsgroups: comp.lang.ada

[…]

I thought of a follow up question…

My understanding is that SPARK assumes there are no flow errors when creating verification conditions. Thus, in general, if one accepts certain flow errors would there be a danger that one might subsequently prove a VC that is not really relevant to the program?

From: Alexander Senier <mail@senier.net>
Date: Thu, 28 Oct 2010 18:26:01 +0200
Subject: Re: Array initialization in SPARK
Newsgroups: comp.lang.ada

[…]

In Section 7.1 (and particularly in 7.1.1 "Array initialization in a loop") of the SPARK Proof manual this issue is discussed. When generating RTCs the Examiner *assumes* that all elements of an array are defined and within their subtype ranges. The best way to ensure that indeed is making the Examiner to show the absence of data flow errors.

If you must initialize arrays in a loop (as in your example), you have to accept the flow errors and review the code to initialize all array elements correctly. You definitely want to be very careful with those accept statements and really simplify your other initialization functions to make that as obvious as possible.

It may be beneficial to restrict the initialization to simple patterns that could be verified (semi-)automatically in the future. E.g. have a for loop over the complete array index and only use its loop variable to index into the array on the left-hand side of an assignment. If the array is not referenced on the right-hand side (or only indices smaller than the current value of the loop variable), all array elements should be valid after execution of that loop. But your problem may not fit into that pattern…

From: Phil Thornley <phil.jpthornley@gmail.com>
Date: Thu, 28 Oct 2010 09:38:40 -0700 PDT
Subject: Re: Array initialization in SPARK
Newsgroups: comp.lang.ada

[…]

> It may be beneficial to restrict the initialization to simple patterns that could be verified (semi-)automatically in the future.

[…]

That's excellent advice.

But if you can't do that, and it really is important to ensure complete initialization, and you are completing run-time check proofs, then you could force a check on the array:

```
--# check for all I in Natural range
                Out_Matrix'Range(1) =>
--#       (for J in Natural range
                Out_Matrix'Range(2) =>
--#          (Out_Matrix(I, J) in
                 Types.Float8));
```

but, for a complex initialization, completing the proof of this check will be correspondingly complex :-(

Conference Calendar

Dirk Craeynest
K.U.Leuven. Email: Dirk.Craeynest@cs.kuleuven.be

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.

2011

January 17-20 — 9th **Australasian Symposium on Parallel and Distributed Computing** (AusPDC'2011), Perth, Australia. Topics include: Multicore; Middleware and tools; Performance evaluation; Parallel programming models, languages and compilers; Runtime systems; Reliability, security, and dependability; Applications; etc.

☺ January 26-28 — 38th ACM SIGPLAN-SIGACT **Symposium on Principles of Programming Languages** (POPL'2011), Austin, Texas. Topics include: all aspects of programming languages and systems, with emphasis on how principles underpin practice.

January 24-25 — ACM SIGPLAN **Workshop on Partial Evaluation and Program Manipulation** (PEPM'2011).

☺ January 29 — 5th ACM SIGPLAN **Workshop on Programming Languages meets Program Verification** (PLPV'2011). Topics include: research at the intersection of programming languages and program verification; attempts to reduce the burden of program verification by taking advantage of particular semantic or structural properties of the programming language; all aspects, both theoretical and practical, of the integration of programming language and program verification technology.

☺ February 09-10 — 3rd **International Symposium on Engineering Secure Software and Systems** (ESSoS'2011), Madrid, Spain. Topics include: security architecture and design for software and systems; verification techniques for security properties; systematic support for security best practices; programming paradigms for security; processes for the development of secure software and systems; etc.

February 09-11 — 19th Euromicro **International Conference on Parallel, Distributed and network-based Processing** (PDP'2011), Ayia Napa, Cyprus. Topics include: Parallel Computer Systems (embedded parallel and distributed systems, fault-tolerance, multi/many core systems, GPU and FPGA based parallel systems ...); Models and Tools for Parallel Programming Environments; Advanced Applications (multi-disciplinary applications, numerical applications with multi-level parallelism, real time distributed applications, distributed business applications ...); Languages, Compilers and Runtime Support Systems (task and data parallel languages, object-oriented languages, dependability issues, scheduling, ...); etc.

☺ February 21-25 — **Software Engineering 2011** (SE'2011), Karlsruhe, Germany. Theme (in German): Ingenieurmäßige Software-Entwicklung für kritische Anwendungen.

March 01-04 — 15th **European Conference on Software Maintenance and Reengineering** (CSMR'2011), Oldenburg, Germany. Topics include: Tools, formalisms and methods supporting the evolution of software systems, software architectures and software models; Methods and techniques to assess, enable, improve and certify maintainability and evolvability of software intensive systems; Experience reports on maintenance and reengineering of large-scale systems; Empirical studies in software reengineering, maintenance, evolution and renovation; Education-related issues to evolution, maintenance and reengineering; Language support for software maintenance and evolution; etc.

☺ March 09-12 — 42nd ACM **Technical Symposium on Computer Science Education** (SIGCSE'2011), Dallas, Texas, USA.

March 14-18 **Design, Automation and Test in Europe** (DATE'2011), Grenoble, France. Topics of Track on Real-time, Networked and Dependable Systems include: real-time programming languages and software, worst case execution time analysis, verification, tools and design methods, dependable systems, software for safety critical systems, etc.

March 21-25 26th ACM **Symposium on Applied Computing** (SAC'2011), TaiChung, Taiwan.

☺ Mar 21-25 **Track on Object-Oriented Programming Languages and Systems** (OOPS'2011). Topics include: Language design and implementation; Type systems, static analysis, formal methods; Integration with other paradigms; Aspects, components, and modularity; Distributed, concurrent or parallel systems; Interoperability, versioning and software adaptation; etc.

☺ Mar 21-25 **Track on Programming Languages** (PL'2011). Topics include: Compiling Techniques, Formal Semantics and Syntax, Garbage Collection, Language Design and Implementation, Languages for Modeling, Model-Driven Development and Model Transformation, New Programming Language Ideas and Concepts, Practical Experiences with Programming Languages, Program Analysis and Verification, Programming Languages from All Paradigms, etc.

☺ Mar 21-25 **Track on Real-Time Systems** (RTS'2011). Topics include: all aspects of real-time systems design, analysis, implementation, evaluation, and case-studies; including scheduling and schedulability analysis; worst-case execution time analysis; modeling and formal methods; validation techniques; reliability; compiler support; component-based approaches; middleware and distribution technologies; programming languages and operating systems; embedded systems; etc.

March 21-25 4th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2011), Berlin, Germany. Topics include: Domain specific testing including, but not limited to, security testing, embedded software testing, OO software testing, ...; Verification & validation; Quality assurance; Empirical studies; Inspections; Tools; Novel approaches to software reliability assessment; etc.

March 21-25 10th **International Conference on Aspect-Oriented Software Development** (AOSD'2011), Porto de Galinhas, Pernambuco, Brazil. Topics include: new notions of modularity in computer systems, software engineering, programming languages, and other areas; Software development methods, Programming language design, Compilation and interpretation, Verification and static program analysis, Refactoring, Distributed/concurrent systems, etc. Deadline for submissions: February 28, 2011 (student forum and poster event).

Mar 26 – Apr 03 **European Joint Conferences on Theory and Practice of Software** (ETAPS'2011), Saarbrücken, Germany.

March 26 11th **International Workshop on Language Descriptions, Tools and Applications** (LDTA'2011). Topics include: program analysis, transformation, generation, and verification; reverse engineering and re-engineering; refactoring and other source-to-source transformations; language definition and language prototyping; debugging, profiling, IDE support; etc.

March 27 3rd **Workshop on Generative Technologies** (WGT'2011). Topics include: Generative programming, metaprogramming; Analysis of language support for generative programming; Case Studies and Demonstration Cases; etc.

April 01-02 9th **Workshop on Quantitative Aspects of Programming Languages** (QAPL'2011). Topics include: probabilistic, timing and general quantitative aspects in Language design, Language extension, Multi-tasking systems, Language expressiveness, Verification, Semantics, Time-critical systems, Safety, Embedded systems, Program analysis, Risk and hazard analysis, Scheduling theory, Distributed systems, Model-checking, Security, Concurrent systems, etc. Deadline for submissions: January 24, 2011 (presentation reports).

April 02-03 8th **International Workshop on Formal Engineering approaches to Software Components and Architectures** (FESCA'2011). Topics include: Interface compliance

and contractual use of components; Modelling formalisms for the analysis of concurrent systems assembled of components; Techniques for prediction and formal verification of system properties, including static and dynamic analysis; Industrial case studies and experience reports; etc.

☺ Mar 28-31 — 14th IEEE **International Symposium on Object/component/service-oriented Real-time distributed Computing** (ISORC'2011), Newport Beach, California, USA. Topics include: Programming and system engineering (ORC paradigms, languages, RT Corba, UML, model-driven development of high integrity applications, specification, design, verification, validation, testing, maintenance, system of systems, etc.); System software (real-time kernels, middleware support for ORC, extensibility, synchronization, scheduling, fault tolerance, security, etc.); Applications (embedded systems (automotive, avionics, consumer electronics, etc), real-time object-oriented simulations, etc.); System evaluation (timeliness, worst-case execution time, dependability, fault detection and recovery time, etc.); ...

☺ April 10-13 — 6th **European Conference on Computer Systems** (EuroSys'2011), Salzburg, Austria. Topics include: all areas of operating systems and distributed systems, including systems aspects of Dependable computing and storage, Distributed computing, Parallel and concurrent computing, Programming-language support, Real-time and embedded computing, Security, etc.

April 12-15 — 2nd **International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering** (PARENG'2011), Ajaccio, Corsica, France.

April 18-20 — 3rd **NASA Formal Methods Symposium** (NFM'2011), Pasadena, California, USA. Topics include: Theorem proving, Model checking, Static analysis, Dynamic analysis, Model-based development, Application experiences, etc.

April 20-24 — 17th **International Symposium on Formal Methods** (FM'2011), Limerick, Ireland. Theme: "Formal Methods Come of Age". Topics include: advances and maturity in formal methods research, education, and deployment via tool support and industrial best practice, and their role in a variety of industries, domains, and in certification and assurance; in particular experience with practical applications of formal methods in industrial and research settings, experimental validation of tools and methods as well as construction and evolution of formal methods tools. Deadline for submissions: January 10, 2011 (papers), January 24, 2011 (tutorials, workshops).

☺ April 25-29 — 5th **Latin-American Symposium on Dependable Computing** (LADC'2011), São José dos Campos, São Paulo, Brazil. Topics include: Dependability of software (frameworks and software architectures for dependability, model driven dependability engineering, testing, verification, software certification, ...); Dependability of maintenance; Dependability and human issues; Security; Safety (safety-critical applications and systems, ...); etc. Deadline for submissions: February 4, 2011 (industry track, fast abstracts, student forum).

April 27-29 — 18th Annual IEEE **International Conference and Workshops on the Engineering of Computer Based Systems** (ECBS'2011), Las Vegas, Nevada, USA. Theme: "Engineering Next Generation Systems". Topics include: Component-Based System Design; Design Evolution; Distributed Systems Design; ECBS Infrastructure (Tools, Environments); Education & Training; Embedded Real-Time Software Systems; Integration Engineering; Model-Based System Development; Modelling and Analysis of Complex Systems; Open Systems; Reengineering & Reuse; Reliability, Safety, Dependability, Security; Standards; Verification & Validation; etc. Deadline for early registration: February 22, 2011.

April 27-29 — 16th Annual IEEE **International Conference on the Engineering of Complex Computer Systems** (ICECCS'2011), Las Vegas, Nevada, USA. Topics include: Verification and validation, Model-driven development, Reverse engineering and refactoring, Design by contract, Agile methods, Safety-critical & fault-tolerant architectures, Real-time and embedded systems, Tools and tool integration, Industrial case studies, etc. Deadline for early registration: February 22, 2011.

☺ May 16-20 — 25th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2011), Anchorage, Alaska, USA. Topics include: all areas of parallel and distributed processing, such as: Parallel and distributed algorithms; Applications of parallel and distributed computing; Parallel and distributed software, including parallel and multicore programming languages and compilers, runtime systems, middleware, libraries, parallel programming paradigms, programming environments and tools, etc.

☺ May 20 12th **International Workshop on Parallel and Distributed Scientific and Engineering Computing** (PDSEC-11). Topics include: parallel and distributed computing techniques and codes; practical experiences using various parallel and distributed systems; task parallelism; scheduling; compiler issues for scientific and engineering computing; scientific and engineering computing on parallel computers, multicores, GPUs, FPGAs, ...; etc.

☺ May 20 **Workshop on Multithreaded Architectures and Applications** (MTAAP'2011). Topics include: programming frameworks in the form of languages and libraries, compilers, analysis and debugging tools to increase the programming productivity. Deadline for submissions: January 10, 2011.

☺ May 21-28 33rd **International Conference on Software Engineering** (ICSE'2011), Waikiki, Honolulu, Hawaii, USA. Theme: "Software by Design". Topics include: Engineering of distributed/parallel software systems; Engineering of embedded and real-time software; Engineering secure software; Patterns and frameworks; Programming languages; Reverse engineering and maintenance; Software architecture and design; Software components and reuse; Software dependability, safety and reliability; Software economics and metrics; Software tools and development environments; Theory and formal methods; etc. Deadline for submissions: January 30, 2011 (student volunteers).

May 21-22 8th **International Working Conference on Mining Software Repositories** (MSR'2011). Topics include: Mining of repositories across multiple projects; Characterization, classification, and prediction of software defects based on analysis of software repositories; Search techniques to assist developers in finding suitable components and code fragments for reuse, and software search engines; Analysis of change patterns and trends to assist in future development; Case studies on extracting data from repositories of large long-lived and/or industrial projects; etc. Deadline for submissions: January 26, 2011 (research/short papers).

May 21-28 2nd **Student COntest on softwaRe Engineering** (SCORE'2011). Deadline for submissions: January 15, 2011 (summary reports), February 28, 2011 (final deliverable), ICSE 2011 (finals).

May 22-24 24th IEEE-CS **Conference on Software Engineering Education and Training** (CSEET'2011), Waikiki, Honolulu, Hawaii, USA. Topics include: Technology Transfer, Student projects and internships, Industry-academia collaboration models, Software engineering professionalism, Education & training for "real-world" Software Engineering practices, Training models in industry, Systems and Software Engineering, etc. Deadline for submissions: January 17, 2011 (panel sessions, practice and methods presentations, posters, tutorials).

May 24-27 22nd IEEE **International Symposium on Rapid System Prototyping** (RSP'2011), Karlsruhe, Germany. Topics include: Real Time embedded system challenges; Specification and Language Models for hardware/software systems; Very large scale system engineering; Embedded System verification/validation; Critical Embedded Systems design; Reliability and failure analysis; Emerging Technologies and Applications; Industrial Designs in Automotive, Medical and Avionics domains; etc. Deadline for submissions: January 16, 2011 (full papers).

☺ May 26-28 9th IEEE **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2011), Busan, Korea. 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, High-performance scientific and engineering computing, etc.

June 01-03 11th **International Conference on Computational Science** (ICCS'2011), Tsukuba, Japan. Topics include: recent developments in methods and modelling of complex systems for diverse areas of science, advanced software tools, etc. Deadline for submissions: January 8, 2011 (full papers). Deadline for early registration: March 31, 2011.

June 06-09 6th **International Federated Conferences on Distributed Computing Techniques** (DisCoTec'2011), Reykjavik, Iceland. Includes the COORDINATION, DAIS, and FMOODS & FORTE conferences. Deadline for submissions: February 6, 2011 (abstracts), February 13, 2011 (papers).

June 15-17 — 36th Annual **USENIX Technical Conference** (USENIX ATC'2011), Portland, Oregon, USA. Topics include: Distributed and parallel systems; Embedded systems; Reliability, availability, and scalability; Security, privacy, and trust; etc. Deadline for submissions: January 12, 2011.

June 20-23 — 2011 **International Conference for Computational Science and its Applications** (ICCSA'2011), Santander, Spain. Topics include: Parallel and Distributed Computing, Security Engineering, Risk Analysis, Reliability Engineering, Software Engineering, etc. Deadline for early registration: April 4, 2011.

♦ June 20-24 — 16th **International Conference on Reliable Software Technologies - Ada-Europe'2011**, Edinburgh, UK. Organized together with the Ada Conference UK 2011, under the common name of "The Ada Connection". Sponsored by Ada-Europe, in cooperation with ACM SIGAda. Deadline for submissions: January 8, 2011 (industrial presentations).

June 20-24 — 9th **Working IEEE/IFIP Conference on Software Architecture** (WICSA'2011), Boulder, Colorado, USA. Topics include: Software architecture and software qualities; Architecture description languages and model driven architecture; Software architecture discovery and recovery; Software architects' roles and responsibilities; Training, education, and certification of software architects; Industrial experiments and case studies; etc. Deadline for submissions: February 7, 2011 (papers).

June 20-24 — 11th **International Conference on Application of Concurrency to System Design** (ACSD'2011), Kanazawa, Japan. 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. Deadline for submissions: January 10, 2011 (abstracts), January 17, 2011 (papers).

June 27-29 — 16th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2011), Darmstadt, Germany.

☺ June 28-30 — 49th **International Conference Objects, Models, Components, Patterns** (TOOLS Europe'2011), Zurich, Switzerland. Topics include: Applications to safety- and security-related software; Distributed and concurrent object systems; Domain specific languages and language design; Experience reports, including efforts at standardisation; Language implementation techniques, compilers, run-time systems; Multicore programming, models and patterns; Object technology, including programming techniques, languages, tools; Practical applications of program verification and analysis; Real-time object-oriented programming and design; Tools and frameworks for supporting model-driven development; Trusted and reliable components; etc. Deadline for submissions: January 28, 2011 (papers).

July 05-07 — 15th **International Conference on System Design Languages of the SDL Forum Society** (SLD'2011), Toulouse, France. Topics include: Industrial application reports (industrial usage and experience reports, tool engineering and frameworks, domain-specific applicability, such as aerospace, automotive, control, ...); Evolution of development tools and languages (domain-specific profiles and extensions, modular language design, semantics and evaluation, methodology for application, standardization activities); Modeling in multi-core and parallel applications; Education and Promotion of System Design Languages; etc. Deadline for submissions: February 1, 2011 (abstracts), February 14, 2011 (papers), May 15, 2011 (posters, exhibits).

July 13-14 — 11th **International Conference on Quality Software** (QSIC'2011), Madrid, Spain. Topics include: Software quality (review, inspection and walkthrough, reliability, safety and security, ...); Evaluation of software products and components (static and dynamic analysis, validation and verification); Economics of software quality; Formal methods (program analysis, model construction, ...); Applications (component-based systems, distributed systems, embedded systems, enterprise applications, information systems, safety critical systems, ...); etc.

July 14-20 — 23rd **International Conference on Computer Aided Verification** (CAV'2011), Snowbird, Utah, USA. Topics include: Algorithms and tools for verifying models and implementations, Program analysis and software verification, Verification methods for parallel and concurrent hardware/software systems, Applications and case studies, Verification in industrial practice, etc. Deadline for submissions: January 14, 2011 (abstracts), January 21, 2011 (papers).

☺ July 25-29 — 25th **European Conference on Object-Oriented Programming** (ECOOP'2011), Lancaster, UK. Topics include: all areas of object technology and related software development technologies, such as Analysis and design methods and patterns; Distributed, concurrent, real-time systems; Language design and implementation; Modularity, components, services; Software development environments and tools; Type systems, formal methods; Compatibility, software evolution; etc.

August 15-18 — 6th IEEE **International Conference on Global Software Engineering** (ICGSE'2011), Helsinki, Finland. Topics include: Strategic issues in distributed development (cost-benefit-risk analysis, ...); Methods and tools for distributed software development (requirements engineering, design, coding, verification, testing and maintenance, development governance); Empirical studies and lessons learnt from distributed development; etc. Deadline for submissions: February 14, 2011 (abstracts, workshops), February 28, 2011 (papers), March 14, 2011 (other contributions).

Aug 29 – Sep 02 — 15th IEEE **International Enterprise Computing Conference** (EDOC'2011), Helsinki, Finland. Topics include: the full range of engineering technologies and methods contributing to intra- and inter-enterprise distributed application systems; industry specific solutions, e.g. for aerospace, automotive, finance, logistics, medicine and telecommunications; etc. Deadline for submissions: February 15, 2011 (abstracts), February 28, 2011 (full papers), March 15, 2011 (workshop papers).

☺Aug 30 – Sep 02 — **International Conference on Parallel Computing** 2011 (ParCo'2011), Gent, Belgium. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments, in particular Applications of multicores, GPU-based applications, Parallel programming languages, compilers and environments, Best practices of parallel computing, etc. Deadline for submissions: March 20, 2011 (extended abstracts of papers), March 31, 2011 (proposals for mini-symposia).

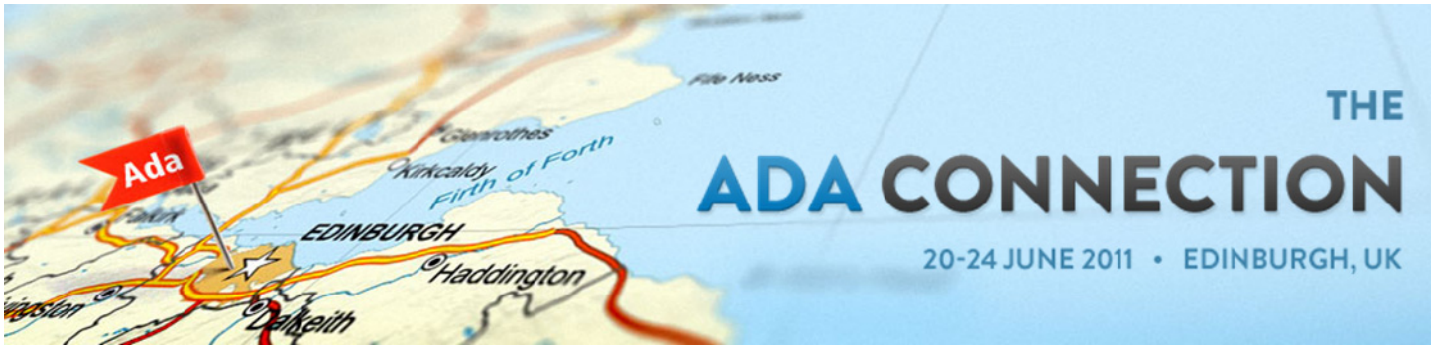
☺ Sep 13-16 — 40th **International Conference on Parallel Processing** (ICPP'2011), Taipei, Taiwan. Topics include: all aspects of parallel and distributed computing, such as Compilers, Programming Models and Languages, Multi-core and Parallel Systems etc. Deadline for submissions: February 24, 2011 (papers).

♦ Sep 14-16 — 15th **International Real-Time Ada Workshop** (IRTAW-15), Liébana (Cantabria), Spain. Deadline for receipt of position paper: 15 May 2011.

September 22-23 — 5th **International Symposium on Empirical Software Engineering and Measurement** (ESEM'2011), Banff, Alberta, Canada. Topics include: Generative programming, metaprogramming; Product-line architectures; Analysis of language support for generative programming; Semantics, type-systems of generative programs; Case Studies and Demonstration Cases; etc. Deadline for submissions: March 15, 2011 (full papers), June 15, 2011 (industry experience reports, short papers, posters).

☺ Sep 26-30 — CBSoft2011 - 15th **Brazilian Symposium on Programming Languages** (SBLP'2011), Sao Paulo, Brazil. Topics include: the fundamental principles and innovations in the design and implementation of programming languages and systems; such as: Programming paradigms and styles, including object-oriented, real-time, multithreaded, parallel, and distributed programming; Program analysis and verification, including type systems, static analysis and abstract interpretation; Programming language design and implementation, including new programming models, programming language environments, compilation and interpretation techniques; etc. Deadline for submissions: April 22, 2011 (abstracts), April 29, 2011 (full papers).

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



The Ada Connection

16th International Conference on Reliable Software Technologies – Ada-Europe 2011

Ada Conference UK 2011

20 - 24 June 2011, Edinburgh, UK

http://www.ada-europe.org/conference2011

Advance Information

The Ada Connection combines the 16th International Conference on Reliable Software Technologies – Ada-Europe 2011 – with Ada Conference UK 2011. It will take place in Edinburgh, Scotland's capital city and the UK's most popular conference destination. Following tradition, the conference will span a full week, including a three-day technical program Tuesday to Thursday, and parallel tutorials and workshops on Monday and Friday. The Ada Connection will also encompass industrial and vendor tracks, and a vendor exhibition on Tuesday and Wednesday, under the banner of Ada Conference UK.

The Ada Connection provides a unique opportunity for interaction and collaboration between academics and industrial practitioners.

About the Venue

The Conference venue is Edinburgh's John McIntyre Conference Centre, a newly extended conference and meeting venue which opened in September 2009. Situated just over a mile away from Edinburgh Castle, the Conference venue is a state-of-the-art, self-contained building with a range of nearby accommodation options and an open-air terrace overlooking Holyrood Park, a public park with volcanic cliff faces whose centrepiece, Arthur's Seat, offers a rewarding climb from which excellent views can be gained of the city and beyond.

Edinburgh is one of the top ten conference destinations in the world, and home to Scotland's five National Galleries and the National Museum of Scotland. Bounded by seven hills and open sea, Edinburgh is an architectural masterpiece; the Old and New Towns are both designated World Heritage Centres. Highlights include the iconic Edinburgh Castle, the stunning Scottish Parliament and the Georgian New Town.

acm In-Cooperation

in cooperation with
ACM SIGAda

Ada europe

CSR
Centre for Software Reliability

THE ADA CONNECTION 2011 EDINBURGH

Invited Speakers

Eminent keynote speakers have been selected to open each of the three days of the core conference programme. Confirmed speakers are:

- **Peter Bernard Ladkin** (University of Bielefeld CITEC and Causalis Limited), a recognised specialist in system safety. Peter will address concerns about the international standard IEC 61508 and its perceived lack of criteria for critical assessment of objective properties of software developed for safety-critical systems. He will also recount progress on guidelines for German applications which go some way towards redressing this perceived lack, in his talk entitled *Future of Software Safety Standards.*
- **Jeff O'Leary** (US Federal Aviation Administration (FAA)), has more than 18 years of experience in software development, systems acquisition and deployment of large mission critical command and control systems. In his talk *Assuring Software Reliability While Using Web Services and Commercial Products* he will present a government software procurement official's perspective on systems development and quality, and discuss the implications, approach and unique issues of building reliable, trusted web services using commercial products.

STOP PRESS: Pippa Moore (UK Civil Aviation Authority) is confirmed as the third keynote speaker

Tutorials

Attendees will have a varied choice of half-day and full-day tutorials that will be offered on Monday and Friday, either side of the central days of the conference. Tutorials consist of courses given by recognised experts in their respective fields, which deal with up-to-date technologies for the development of reliable software.

Social Programme

The social programme offers a congenial Reception on Tuesday evening and a Conference Banquet on Wednesday evening. The reception will include a "whisky tasting", providing the opportunity to sample a range of specially selected single malt Scotch whiskies. The banquet will be held at the historic and scholarly Signet Library on the Royal Mile in Edinburgh city centre.

Further Information

The conference website at `http://www.ada-europe.org/conference2011` will provide full and up-to-date details of the program, venue and social programme, accommodation and travel advice. For exhibiting and sponsoring details please contact the Exhibition Chair, Joan Atkinson, at Joan.Atkinson@ncl.ac.uk.



15TH INTERNATIONAL REAL-TIME ADA WORKSHOP (IRTAW-15)

September 14-16, 2011 – Liébana (Cantabria), Spain

http://www.artist-embedded.org/artist/IRTAW-15.html

CALL FOR PAPERS

Since the late Eighties the **International Real-Time Ada Workshop** series has provide d 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 and to the proposals for Ada 2012, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar profile.

In keeping with this tradition, and in light of the current revision process that will lead to the new Ada 2012 standard, the goals of **IRTAW-15** will be to:

- review the current status of the Ada 2012 Issues that are related with the support of real-time systems;
- examine experiences in using Ada for the development of real-time systems and applications, especially – but not exclusively – those using concrete implementation of the new Ada 2005 real-time features;
- report on or illustrate implementation approaches for the real-time features of Ada 2012;
- 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 extra-functional concerns;
- examine and develop paradigms for using Ada for real-time distributed systems, with special emphasis on 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;
- 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-15** 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.

Format

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) (to be confirmed). Selected papers will also appear in the Ada User Journal.

Submission

Please submit position papers, in PDF format, to the Program Chair by e-mail: **aldeam@unican.es**

Important Dates

Receipt of Position Paper: 15 May 2011
Final Copy of Paper: 31 July 2011
Notification of Acceptance: 15 June 2011
Workshop Date: 14-16 September 2011

Introduction

"The Ada Way" is an annual student programming contest organized by Ada-Europe, the international organization that promotes the knowledge and use of Ada in European academia, research and industry. A Steering Committee formed by representatives of promoting institutions oversees the organization of the contest. The Steering Committee is currently comprised of: Dirk Craeynest and Ahlan Marriott (Ada-Europe), Ricky Sward (ACM SIGAda), Jamie Ayre and Matteo Bordin (AdaCore), Jean-Pierre Fauche (Atego), Ian Broster (Rapita), Rod White (MBDA).

This initiative aims to attract students and educators to Ada in a form that is both fun and instructive. For this reason the contest is a yearly programming competition among student teams, whereby each team must have a university affiliation and be endorsed by an educator. The ideal, but not exclusive, context for participation is as part of an organized teaching/course activity in which the theme and requirements of the contest are endorsed and supported by the educator. See the "Participation Requirements" section for details.

The contest opens in September with the announcement of the theme, and allows submissions until the end of April the following year. See below for the 2010-11 edition theme and the Submissions section for the submission requirements.

Students and educators who may consider participating and want more information on "The Ada Way" in general and its 2010-11 edition in particular are invited to make contact with the Steering Committee at *board@ada-europe.org*.

Project Theme for Academic Year 2010-11: Software simulator of a football (soccer) match

The following specification intentionally leaves some room for interpretation and extension: participants are encouraged to use their intelligent creativity to firm up the derivative specification they want to work against.

The software system shall support at least the following features:

- Users must be able to play a single game; support for playing a series of matches, with fixtures and associated rules, is optional and can be omitted
- The chosen variant of the game shall be configurable in all relevant parameters, allowing for any of 5-a-side, 7-a-side, and the canonical 11-a-side formats
- The members of the squads will feature individually configurable characteristics for, at least, technical and tactical skills, speed, physical parameters including fatigue; some of those parameters shall be dynamic and evolve with the match according to some programmed logic
- Each squad shall have a (software) manager able to configure the initial players line up, the initial tactic and to issue commands for tactic changes and substitutions, all subject to the rules of the game as in the corresponding standard
- Each squad shall play according to the tactic commanded by the manager; deviations shall be permitted in so far as they result from programmable characteristics of the players
- Each match shall have one independent (software) referee and two to three subordinate (software) assistants who control the game and ensure that the applicable rules are followed; the behavior and the performance of the referee and assistants need not exhibit the physical limitations of actual humans.

The software system shall include at least:

- A software core, whether centralized or distributed, implementing all of the logic of the simulation
- One read-only graphical panel (window) for the display of the football field, the players, the ball, the referee and assistants; as for the (simulated) human figures on the pitch it shall be sufficient to represent them as moving numbered dots on the display without resorting to sophisticated graphical rendering, as in a view of a subbuteo table seen from the top
- Two distinct read-write graphical panels (windows) for the user to influence the otherwise independent action of the team managers; the panel shall display the current parameters for each player; the refresh rate of such display shall be user-configurable
- One read-only graphical panel (window) for the display of a user-configurable selection of statistics; the refresh rate of such display shall be user-configurable.

The software core shall be programmed in Ada. The software design shall permit the principal algoritms to be modified and replaced at will: in other words, the software system shall be as modular, configurable and scalable as possible. These qualities will contribute to the evaluation.

The graphical panels can be programmed in any language that the participating teams will consider fit for purpose. The graphical beauty of such panels will however be only a minor factor in the evaluation. What shall matter instead is that the interaction and the flow of data and control between the software core and the graphical panels is governed by good architectural principles and shows sufficient accuracy and performance.

To be considered for evaluation, the system shall run out of the box. The target platform may be freely chosen between Linux, Windows and MacOS. Portability across them will however be a competitive advantage.

Participation requirements

Participating teams shall be composed by a minimum of 2 and a maximum of 7 members. Each team shall have a codename and a logo. Team work may be performed as part of an organized teaching/course activity or as a volunteer project. Either way, each team must be recognised and endorsed by an academic educator.

Team members must be full-time students: they must provide evidence of their status when submitting their project. The contest is open to undergraduate and Master students. Teams may but need not include a mix of undergraduate and graduate students. Team members may belong to distinct institutions.

Submission

The software system shall be delivered in source (as a single compressed archive), accompanied by:

1. A software specification document (in PDF), which describes the principal design decisions and argues their quality, and presents the points of extension and modification in the system; the specification shall clearly single out all places at which the team made arbitrary interpretation of the specification or added or extended requirements
2. A user manual describing the compilation and installation procedures, the configuration options and the allowable use of the system (in PDF)
3. The team codename, logo and composition: name, email contact, evidence of enrollment as full-time students (in a single PDF)
4. The written endorsement to the submission by an academic or otherwise senior instructor in whose class the project was launched (in PDF).

The submission shall be made as a single compressed archive of all items listed above at the URL that will appear on this page in due time.

All sources shall be released for the good of the general public, to become reference material for educational and promotional purposes. To this end the use of GPL (GNU General Public License) is recommended, though we are not prescriptive of a specific scheme, so long as the general intent of free dissemination is preserved.

Submissions shall be accepted during the whole month of April 2011, at the Ada Way website, *http://www.ada-europe.org/AdaWay*.

Evaluation and Prize

The evaluation criteria will include:

- Coverage of requirements
- Syntatic, semantic, programmatic and design correctness
- Clarity and readability of the code
- Quality of design
- Ingenuity and cuteness of the solution
- Time and space efficiency of the solution.

The evaluation will be performed by a team of distinguished Ada experts comprised of: John Barnes (UK), Tucker Taft (US), Joyce Tokar (US), Pascal Leroy (F), Ed Schonberg (US).

The winning submission shall be announced on 31 May 2011 by a post on the site and by an email communication to all participating teams.

The prize will consist of: a framed award; one free registration and up to 3 reduced student fees for representatives of the winning team to attend to the Ada-Europe 2011 Conference; accommodation and airfare for the team representatives (with ceiling at EUR 3,000); an exhibition slot in the conference program; visibility in electronic and printed media including:

- Ada User Journal: *http://www.ada-europe.org/journal.html*
- Ada Letters: *http://www.sigada.org/ada_letters/*

For up-to-date information on Ada-Europe's student programming contest, please go to the official web site of "The Ada Way", *http://www.ada-europe.org/AdaWay,*

Sponsors

This year's competition is sponsored by Ada-Europe, AdaCore, and Atego.

Overview of the 14th International Real-Time Ada Workshop

7-9 October 2009
Portovenere, Italy

Contents *

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.

Workshop Participants

José Ruiz, AdaCore, France
Edmond Schonberg, AdaCore, USA
Stephen Michell, Maurya Software, Canada
Rod White, MBDA, UK
Kristoffer Nyborg Gregertsen, Norwegian Institute of Science And Technology, Norway
Bjorn Andersson, Polytechnic Institute of Porto, Portugal
António Barros, Polytechnic Institute of Porto, Portugal
Luis Miguel Pinho, Polytechnic Institute of Porto, Portugal
Joyce Tokar, Pyrrhus Software, USA
Juan Antonio de la Puente, Technical University of Madrid, Spain
Juan Zamorano, Technical University of Madrid, Spain
Jorge Real, Technical University of Valencia, Spain
Sergio Saez, Technical University of Valencia, Spain
Mario Aldea Rivas, University of Cantabria, Spain
Michael González Harbour, University of Cantabria, Spain
Javier Gutierrez, University of Cantabria, Spain
Enrico Mezzetti, University of Padua, Italy
Marco Panunzio, University of Padua, Italy
Tullio Vardanega, University of Padua, Italy
Alan Burns, University of York, UK
Abdul Haseeb Malik, University of York, UK
Andy Wellings, University of York, UK
Carl Brandon, Vermont Technical College, USA

Sponsors

* The Proceedings of the 14th International Real-Time Ada Workshop appeared in ACM Ada Letters, Volume XXX, Number 1, April 2010; session summaries reprinted with permission.

Multiprocessor Systems Session Summary

Chairs/ Rapporteurs: Alan Burns and Andy J. Wellings

Abstract

This report summarizes the discussion held at Fourteenth InternationalWorkshop on Real-Time Ada Issues (IRTAW 14) on how to provide better support for multiprocessor systems in Ada.

1 Introduction

The whole first day of the workshop was dedicated to a discussion of multiprocessor issues. The discussions were partitioned into two sub sessions. The first was chaired by Alan Burns and covered the following topics:

- scope of the issues to be addressed;
- current state of real-time multiprocessing scheduling;
- the development of a proposed model for Ada, including both task allocation, protected object access protocols and interrupt handling.

The second session was chaired by Andy Wellings and addressed the following issues:

- Execution-Time Clocks and Timers
- Group Budgets
- Ravenscar Issues
- Non SMP Architectures

Ada 2005 is about to undergo further updates and any proposals for changes have to be raised by the end of October 2009. Consequently, although the overall goal of the sessions was to consider a wide range of issues, it was agreed that the focus should be on developing better support for real-time scheduling on multiprocessor systems.

2 Scope of Issues

Early on in the discussions it was reaffirmed that traditionally the unit of concurrency in Ada had always been coarse grained and expressed via the task construct. Although for high-performance computing there may be some need to be able to express data-level parallelism, the workshop focus was real-time and it was agreed that we should only consider tasks.

It was further agreed that Ada 2005 currently does already support multiprocessor systems and that nothing we proposed should undermine the current specification, unless it was inherently broken. In the absence of any directives in the program, the behaviour of the program should be that currently defined in the Ada Language Reference Manual. The workshop took the Ada model to be essentially the following:

- all tasks executing in a partition must be able to access shared memory; and
- scheduling between tasks in a partition is global, hence when a processor becomes available it must be given to the highest priority runnable task.

For the above reason, the workshop decided to focus on multiprocessor systems that have access to shared memory and are symmetric, i.e. SMP architectures. There was some discussion on whether the IO space of an SMP architecture could be accessed from all processors. It was assumed that in general all memory locations/device registers could be assessed from all CPUs and that only interrupt may be constrained to certain CPUs.

No assumptions were made about the speed of the individual CPUs. Hence, hyperthreading architectures were in scope.

3 State of the Art in Multiprocessor Scheduling

Alan summarized the possibilities for real-time multiprocessor scheduling. The focus was on where tasks can run.

- Fully global partitioning – Any task can run on any of the available CPUs. The scheduler decides order and placement. Typically these require a global ready queue and there are some concerns about whether this can be implemented efficiently and whether it is scalable to large systems.
- Task partitioned – Each task can only run on one CPU, this being specified by the programmer. However, the specified CPU could be changed at run-time.
- Job partitioned – When a task is released it can run on any CPU, however preempted tasks return to the CPU on which they were initially allocated. Hence, the tasks can migrate only at release time.

It was agreed that the state of the art in multiprocessor scheduling was not mature enough to focus on supporting any particular approach. However, it was agreed that some form of both global and partition scheduling is needed. The reason for this is that better schedulability can be obtained by fixing CPU intensive tasks to a single processor and allowing the others to migrate. Also a task may need to be on a particular processor to service a device's interrupts.

4 The Proposed Model

During the next few hours, discussions were held on what was an appropriate model for Ada. The model that emerged can be summarized by the following points:

This paper previously appeared in ACM Ada Letters, Volume XXX, Number 1, April 2010; reprinted with permission.

- The introduction of the notion of an **allocation domain** [1]. In the multiprocessor literature, allocation domains are sometimes called clusters. The key point about an allocation domain is that it defines a set of CPUs on which tasks are globally scheduled.
- There is a *default* allocation domain (the System domain) that is the set of all processors allocated to a program. *The assumption is that this is a fixed set of processors that does not change during the execution of the program.* The default behaviour of an Ada program if it does not specify anything is therefore global scheduling of all tasks across all processors. Depending on the run-time infrastructure, tasks may migrate between processors only at release time (called job-level allocation in Section 3) or within a release (called fully global allocation in Section 3) [2].
- Allocation domains *cannot* overlap; that is no CPU can be in more than one allocation domain. The reason for this constraint is to ease schedulability analysis and to support efficient global run queues and scaling of the system to hundreds of processors.
- The model allows the programmer to constrain a task to execute on only one CPU in an allocation domain. This was seen as more useful than allowing a task to be fixed to a subset of the CPUs in the allocation domain. Also some current real-time scheduling practices require this.

The following issues were also discussed

- **Failure semantics**: Currently Ada has an implicit semantics of crash failures. It was decided that although the workshop would ideally like to support programs executing on platforms with partial failures, this was too big an issue to address within the current time constraints and the absence of any position papers on the topic.
- **Migration of tasks across allocation domain**: Should a task be able to automatically migrate across allocation domains? The main reasons for allowing this might be load shedding during a transient overload, or reconfiguration following a partial failure.The current Ada mechanism for accessing protected objects from multiple CPUs is not fully defined by the language. Instead implementation advice is given. In this, the assumption is that tasks will busy-wait (spin) at their active priorities for the lock (although other implementations will be allowed). This was not changed by the Workshop. It was decided (mainly for simplicity of feasibility analysis) that migration between allocation domains should not be supported but that the domain of a task could be changed by the programmer at run time.

4.1 Protected Objects

There was some discussion on whether protected objects should be given allocation domains and potentially fixed to specific processors in those domains. The main requirement for this was device access. However, it was agreed that in the model under consideration all device registers were accessible for all CPUs and that interrupts, had “affinities” not protected objects.

The rules for setting the ceiling priorities were deemed by the workshop not to be part of the language. However, for completeness they were given.

- For fully global scheduling – setting the ceiling priority of a protected object that is only accessed within a single allocation domain can use the usual approach of setting ceilings to max priority of the accessing tasks plus 1 (note it must be plus 1 for the global scheduling to work).
- Fixed tasks – Where tasks are fixed to a processor in the same allocation domain, care must be taken and the interaction between tasks and protected object must be understood when setting the ceilings. It is probably safest to force non-pre-emptive execution of protected subprograms.
- If the underlying platform only supports job-level scheduling then all protected objects shared across processors should be accessed non-preemptively.
- For protected objects shared between allocation domains, the protected objects must run non pre-emptively. This is because there is no relationship between the priorities in one allocation domain and those in another.
- A lock is always required; using the priority model for locking is not sustainable with multiprocessors (unless it is possible to show that a protected object is only accessed from one processor).

It was also noted that on multiprocessor systems:

- Nested protected object locks can cause deadlock (there are some schemes in the literature to avoid this – for example for each chain another lock must be acquired first)
- Chain blocking is possible.
- In the absence of deadlock, blocking can be bounded.

4.2 Interrupt Handling

In the Ada model, interrupts are mapped to protected procedure calls. Typically these have ceiling set to the hardware priority of the interrupt. The workshop discussed at length how best to ensure mutual exclusive access to interrupt handling protected objects. Various models were considered, including migrating a task to the site where the interrupt is delivered and using the priority model, or disabling/masking the interrupt. In the end it was agreed

[1] We use this term as being short for scheduling or dispatching allocation domain.

[2] During the workshop Björn Anderson considered the job partitioning approach and showed, from a scheduling point of view, that the approach could lead to very poor performance. The workshop therefore restricted its consideration to the task and global partitioning approaches.

that it is the run-time responsibility to ensure mutual exclusion of the protected object in the presence of user tasks calling the procedures.

The workshop did agree however that the 'affinity' of an interrupt should be available to the program so that a 'released' task can be co-located on the same CPU.

5 Execution-Time Clocks and Timers

The workshop discussed whether there were any multiprocessor issues with execution-time clocks and timers.

It was agreed that:

- Measuring execution times presented no additional problems on a multiprocessor as tasks can only be active on one CPU at a time.
- Timers similarly should be no problem. The interrupt from a timer may be constrained to be handled on one processor, but this simply required that the associated PO's subprograms execute non-preemptively.
- *Deciding what the values should be for the WCET on a multiprocessor system is problematic, particularly if processors run at different speeds*. However, this is not a language or a run-time issue. If was suggested that if the programmer was enforcing execution times of a task then its allocation domain should perhaps be set so that the task runs at a uniform speed.

6 Group Budgets

Supporting group budgets in a multiprocessor system is fraught with difficulties. Andy in his overview of the topic gave three approaches:

1. Group budgets can be active on many processors and processors may have variable speed.
2. Group budgets can be active on many processors but processors must have the same speed.
3. Group budgets can only be active on one processor at a time.

The first approach, the workshop felt, would be very difficult to implement accurately without hardware support.

The second approach had two possible implementation models. The first was that at every preemption and release point: the run-time had to look at all running tasks and the group budget for all these tasks. For each group budget, the run-time divides the remaining budget by the number of running tasks and sets timers for this time. When timers go off, the budget has expired.

The second implementation model tries to reduce the run-time cost of the above approach by considering only task release points. When a timer expires in this case, the budget needs to be checked and the timers reset if the remaining budget is greater than 0.

Given the complexity of a multiprocessor group budget, the workshop supported the single-processor group budget approach. It was felt that the use cases for the multiprocessor cases were not clear from a scheduling point of view.

It was noted by the workshop, that the current Ada Reference Manual supported implicitly multiprocessor budgets and that this had to be changed.

7 Non SMP Architectures

Ada was designed to support multiprocessor applications where there is memory shared between each processor. Although the workshop had hoped to discuss non-SMP architectures, it was decided that this was a big issue and left to another day.

8 Ravenscar Issues

For the Ada Ravenscar Profile, the workshop believed that the most restrictive model is probably the best:

- No creation of allocation domains.
- Each tasks fixed to a single processor (either by the programmer, or by the system).
- Interrupts are fixed by the run-time to one of the processors in the system allocation domain.

However, the workshop did feel that having multiple allocation domains was not out of the question.

8.1 Non-multiprocessor Ravenscar issues

Three other non-multiprocessor Ravenscar issues were discussed during the first day of the workshop. One concerned the programming of 'recovery' after an executing-time overrun or a deadline miss. It was felt that a new profile, Ravenscar+, that was still significantly smaller than the full language, was desirable. The definition of such a profile will be discussed at the next workshop.

The second, minor, issue concerned the fact that although relative delays are not permitted in Ravenscar, a relative delay via a timing event was possible. The workshop felt that this bug in the language definition should be fixed.

The final issue concerned timers which are currently excluded from Ravenscar. A number of people felt that as library-level timing events were permitted then timers (restricted to one per task) should also be allowed. There was some concern voiced as to the asynchronous nature of timer events – Ravenscar has eliminated most such events. A vote showed a majority in favour of a change to the definition of Ravenscar to include timers.

Session Summary: Language and Distribution Issues

Chairs: Tullio Vardanega and Michael González-Harbour

Rapporteur: Luís Miguel Pinho

1 Introduction

The goal of the session was to consider a set of additions and changes to the language arising from the accepted position papers, but still not consensual. The session lasted a full day, with Tullio Vardanega and Michael González-Harbour in charge, respectively, of the Morning and Afternoon periods.

At the beginning of the day, Tullio started by presenting the outline of the session, with an initial list of issues to discuss but also noting that the agenda was to be regarded as fairly open and that new or returning issues could easily be integrated.

The anticipated topics (and proponents) for the session were:

- Non-preemptive scheduling and the use of Ravenscar and sporadic tasks with EDF, proposed by Rod White;
- Named memory (storage) pools, non-blocking delays for hardware interfacing and parallel release of barriers, proposed by Luke Wong, Stephen Michell and Brad Moore;
- Generalising EDF support and user-defined clocks, by Andy Wellings and Alan Burns;
- Execution-time accounting of interrupts handlers, a topic with two position papers, one by Mario Aldea Rivas and Michael González-Harbour, and another by Kristoffer Nyborg Gregertsen and Amund Skavhaug;
- The Real-Time Transaction model, by Héctor Pérez Tijero, J. Javier Gutiérrez and Michael González-Harbour.

Other issues were also discussed during the session, either because they were continued from the previous day, or because they were considered to be related:

- Non-Uniform Memory Access architectures, by Andy Wellings, Abdul H. Malik, Neil Audsley and Alan Burns;
- The Ravenscar profile and Multiprocessors, by José Ruiz;
- The Real-Time Framework, by Jorge Real and Alfons Crespo.

The session was highly dynamic, with several rounds of discussion. In order to increase the readability of this report, the successive rounds of discussion on individual topics are collated in a single presentation, instead of actually following the chronological flow of the session.

2 Discussion

2.1 Non-preemptive scheduling

This topic started with a presentation by Rod White, proposing extensions to the Ada real-time features, mainly motivated by experience on practical industrial use of (some of the) new capabilities of Ada 2005 [1]. Rod's objectives were also to make the new features easy to use, without disrupting the current ones, introducing more transparency, as complexity or obscurity would impair industrial adoption.

In particular, Rod introduced three topics for discussion [1]:

1. A more complete and transparent model for the control of dispatching points in non-preemptive scheduling;
2. The use of the Ravenscar profile in conjunction with EDF dispatching;
3. The treatment of sporadic tasks under EDF dispatching.

In the first topic, Rod noted that the current mechanism for tasks to yield the processor in the non-preemptive model is to perform a non-blocking delay, such as a delay into the past (`delay until Clock_First`), which, although giving a correct execution, does not provide a clear understanding.

Furthermore, when a task relinquishes the processor, it is placed at the tail of the ready queue for its priority. Therefore, this command cannot be executed inside a protected subprogram. Rod then proposed a new package for handling non-preemptive scheduling:

```
package Ada.Dispatching.Non_Preemptive is
procedure Yield_To_Higher;
procedure Yield; -- Bounded error if executed within a
                    protected operation
end Ada.Dispatching.Non_Preemptive;
```

`Yield` would replace the non-blocking delay as the mechanism for relinquishing the processor. `Yield_To_Higher` would put the task at the head of its priority queue, therefore only allowing higher priority tasks to execute. This would allow it to be performed inside protected subprograms.

Alan Burns then noted that `Yield` could have a different behaviour inside a protected subprogram, deferring the

This paper previously appeared in ACM Ada Letters, Volume XXX, Number 1, April 2010; reprinted with permission.

Yield operation to when the task left the protected object, instead of giving a bounded error.

In addition to the new package, Rod also proposed a new pragma (Cooperate_On_Priority_Change), that would allow the runtime to implicitly perform Yield_To_Higher operations anytime a task's priority was changed (such when entering or leaving a protected object). This would permit a more precise control on the responsiveness of the system.

In a second round of discussion, the workshop came back to this issue, and a proposal was made to include Yield and Yield_To_Higher in the language. The actual wording would be worked offline, but the general model was considered OK. A straw vote was taken, and the proposal was accepted with 19 votes in favour and 3 abstentions.

As for the Cooperate_On_Priority_Change pragma, there was a consensus that this was a new scheduling model and that as such it needed further thoughts. The proposal was thus withdrawn.

2.2 Ravenscar and EDF

On a different topic, Rod noted that EDF allows higher levels of processor utilisation, thus proposed its use within the Ravenscar profile, by introducing a second parameter to the pragma Profile:

```
pragma Profile (Ravenscar, EDF_Across_Priorities);
```

that would default to current behaviour (FIFO_Within_Priorities), if not specified. This would also allow for other scheduling strategies (such as non-preemptive) to be used together with Ravenscar.

It was not clear to Rod if this would be a variant of Ravenscar, or a new profile. Creating a new profile could be heavy for a simple change in the dispatching policy. Furthermore, Ravenscar is well established in industry – a new name would not be regarded as highly. Therefore, Rod's proposal was to maintain the Ravenscar brand, although probably with some "EDF tag".

At this point Tullio Vardanega asked whether the proposal also considered priority bands or only a model where all priorities had the same dispatching policy. Rod confirmed that his idea was the latter. Michael González-Harbour then noted that by allowing the use of priority bands would permit to mix critical and non critical tasks in the same system, thus with higher levels of flexibility. Nevertheless, Rod answered that for simplicity they would not implement priority bands in their Ravenscar kernel. Rod put forward that the Ravenscar kernel is used in non-critical parts of the system, for efficiency reasons, where EDF could be usefully employed.

In a second round of discussion, Joyce Tokar presented a model of "overriding" the dispatching policy in Ravenscar. Nevertheless, it was noted that all of the mechanisms which are used for EDF control would need to be analysed considering the Ravenscar restrictions (e.g. changing deadlines). The group considered that further work was required to think this proposal through and recommended this to be a subject for the next workshop.

2.3 Sporadic tasks under EDF

In another topic, also presented by Rod, Ada's implementation of EDF dispatching assumes that tasks are periodic, and are scheduled using the clock. However, in embedded systems there are cases where periodic events in nature are released by an external event, and not by the language clock. And, for this case, there is no equivalent to the procedure Delay_Until_And_Set_Deadline, thus it is not possible to set the deadline of sporadic tasks when they are suspended.

Solutions to this problem are possible with the current features of the language, based on protected objects (examples of this are presented in [1]). However, a more efficient solution, and equivalent to the periodic Delay_Until_And_Set_Deadline mechanism, would be an extension to the Suspension_Object, through a child package:

```
package Ada.Synchronous_Task_Control.EDF is
  procedure Suspend_Until_True_And_Set_Deadline (
        S : in out Suspension_Object;
        D : in Ada.Real_Time.Time_Span);
end Ada.Synchronous_Task_Control.EDF;
```

At this point, Andy Wellings raised the issue that user defined clocks (which would be discussed later on) could be a potential solution to this problem. However, it was not clear how that would be accomplished.

A vote was taken later in the session, and the proposal received 8 votes in favor and 12 abstentions. In the Workshop's tradition, abstentions are taken as "informed non-opposition", thus this result was a sufficient basis to promote an AI on the topic. Alan then accepted to prepare an AI to support the proposal.

Another issue was raised with the recurrent use of the "Suspend/Delay until true and set something", since in the previous day the same model was proposed for setting processor affinities. This was regarded as not scalable, as the number of attributes to set when suspending is increased. Although no conclusion was reached on the issue, it was considered something that should be further analyzed.

2.4 Non-blocking delays for hardware interfacing

After Rod's presentation, the session moved on to Stephen Michell, who presented for consideration in the session [2,3]:

- Non-blocking delays for hardware interfacing
- Named memory (storage) pools
- Parallel release of barriers

In the first item, Steve presented the problem when a task needs to access hardware, but requires it to be ready after some "settle" time. The solution for the task to suspend itself is not possible inside a protected object (potentially

suspending operations are not permitted), but if the task spin-waits it consumes CPU time undesirably.

The proposal was then to allow for some kind of delay, inside a protected subprogram. A consequence would be that objects would have to implement actual locks, which is not a problem in multiprocessor systems, where priority-based locks are no longer effective.

Alan Burns then put forward that the same effect could be currently obtained by setting a timer, and requeuing to a private entry. Upon expiration of the timer, the task would be released and could access the hardware. After some consideration, this was accepted as an already existent solution, and the proposal was withdrawn.

2.5 Named memory (storage) pools

Steve also presented a proposal to give the programmer more control on the specification of storage pools. In order to be able to specify and interface with different types of memory, a new API was proposed, where the memory model became similar to direct file I/O [3].

The proposed API would allow to create, read and write memory, but with a more precise control of the characteristics of the underlying memory type (for instance read only). The dynamic addition of memory to the pool was also analysed but considered more complicated to implement.

Andy Wellings also presented his view that many times it is necessary to be able to specify the actual address of storage pools. An attribute 'Address could simply solve the problem. Steve noted that it would not be enough, as special memory may require explicit reads or writes to be performed.

Andy also added that in Non Uniform Memory Access (NUMA) architectures the same problem exists, since it is necessary to separate and identify the heap of each processor. There was some discussion on this issue, but no conclusion was drawn.

2.6 Parallel release of barriers

Another proposal by Steve was to allow the parallel release of several tasks in the same event, which would be useful in a multiprocessor environment. Currently, Ada defines that only the task at the head of an entry queue is released, which means that the release of multiple tasks must be sequential. Also, suspension objects only allow one suspended task.

Steve then proposed to add parallel release capabilities to protected entries and suspension objects. The first would be provided by a pragma Barrier_Entry which would identify an entry as a barrier. Furthermore, to allow for parallel execution inside the entry, it would not be allowed to change the protected state.

The latter would be supported by adding a new type Group_Suspension_Object to Ada.Synchronous_Control, which would allow one to specify a maximum count of suspended tasks. This solution would be simpler, but less flexible. Protected objects are general programming constructs, therefore a solution for protected entries must be much more generic, thus complex.

A note was then made that whatever the solution, it should not break if implemented in a sequential environment (e.g. uniprocessor systems). The resulting behaviour would need to be the same.

Ed Schonberg presented his view that, in order to avoid confusion with regular entries, this mechanism would need its own syntax. This was generally agreed upon by participants, but two different perspectives were put forward: either to add a barrier condition to functions, or to create "entry functions". The advantage of the last is that the specification of the protected object would clearly indicate the parallel release. The existence of a barrier condition in the function would only be known within the protected body.

Later in the session, and after some work during the break, Steve proposed an API for the suspension barrier and a model for "entry functions".

For the suspension barrier, the new proposal was not to mix barriers with suspension objects, and thus implement a child package instead:

```
package Ada.Synchronous_Control.Suspension_Group is
  type Group_Suspension_Object_Status is record
    Last_Released : Boolean;
    Count         : Positive);
  type Group_Suspension_Object(Count : Positive) is
limited private;
     -- count of 1 => unlimited blocking count
     -- and explicit release
     -- suspended tasks can always be released by
     -- call to Set_True
  procedure Set_True (
       S : in out Group_Suspension_Object);
  procedure Set_False (
       S : in out Group_Suspension_Object);
  procedure Current_Status(
       S : Group_Suspension_Object)
    return Group_Suspension_Object_Status;
  procedure Suspend_Until_True(
       S : in out Group_Suspension_Object;
       Unique : out Boolean);
end Ada.Synchronous_Control.Suspension_Group;
```

It was generally agreed that this could be accepted, but it was anyhow decided to defer the issue to the next day for further iteration.

Concerning the protected object model, the proposal was to support special "entry functions", where the last released task would need to set the barrier back to false. This means that this special task would need to change the state of the protected object, which could impact the other tasks executing in parallel inside the entry. The solution was to define a pragma Modifiable_State, which permits to specify what state of the object could be changed by the last released task. All other tasks would not be able to read it.

```
protected type PT is
   ...
   entry function Barrier( A, B, C : Integer ) return Integer;
   pragma Barrier_Entry( Entry_Name => Barrier,
                         Count => N);
   procedure Release;
private
   State : Local_State;
   Go_Now : Boolean;
   Total_Count : Integer;
   pragma Modifiable_State(Total_Count);
end PT;
protected body PT is
   entry function Barrier( A, B, C : Integer ) return Integer
      increments Total_Count when Go_Now is
   begin
      . . .
      if PT'Last_Released then
         Total_Count := 0;
      end if;
   end Barrier;
   procedure Release is
   begin
      if Total_Count = N then
         Go_Now := True;
      or else Now then
         Go_Now := True;
      end if;
   end release;
end PT;
```

Ed Schonberg noted the drawback of the model where one particular task (the last released one) has a special role, as it must be explicitly handled by the programmer inside the entry code. Some proposals were then made to include this special code inside the when clause of the entry, or it to be handled by the releasing task (by requeuing and waiting for the last task to leave the entry function).

A question was asked whether the ARG would accept a change of syntax, as this was the first proposal within the workshop with that requirement. It was considered possible by the members of the ARG present in the meeting, as this was a localized change.

The general feeling was that, although interesting, the model of how to control the resetting of the barrier after all task were released was still not yet mature enough.

The workshop came back to this issue in a third round of discussion, with a set of different proposals made by Andy Wellings. In the first proposal, and considering that the guard is not re-evaluated after each task release (tasks are all released simultaneously), it was up to the releaser to wait to clean the object state:

```
entry Go is
begin
  Data_Available  := True;
  if Parallel'Count > 0 then requeue Clean_Up2;
  else requeue Clean_Up1;
  end if;
end;
entry function Parallel when Data_Available and
       Parallel'Count = 10 is
begin
  return Data;
end;
entry Clean_Up1 when Parallel'Count >  0 is
begin
  requeue Clean_Up2;
end;
entry Clean_Up2 when Parallel'Count = 0 is
begin
end;
```

A problem existed however, if a new task calls function parallel while the other task is executing inside. That needed further consideration.

The second approach was to add a special entry'completion procedure to be executed after all tasks are released:

```
entry Go is
begin
  Data_Available  := True;
end;
entry function Parallel when Data_Available  and
       Parallel'Count = 10 is
begin
  return Data;
end;
when Parallel'Completion  procedure Clean_Up  is
begin
end;
```

The third proposal was to create a special finalize block that could be executed only once and that could modify the state of the object:

```
entry Go is
begin
  Data_Available  := True;
end;
entry function Parallel when Data_Available and
       Parallel'Count = 10 is
begin
  return Data;
finally
  -- can update state
  -- only executed once
end;
```

No decision was eventually made, and the issue was deferred.

2.7 User-defined clocks

The workshop then addressed a proposal [4] from Andy Wellings and Alan Burns to revisit user-defined clocks, something that was considered for Ada 95 but that was not included in the language. Examples exist of a variety of different clocks in embedded systems (for example, a GPS clock), and Ada currently allows for implementations to define other clocks and make them available to

applications. Nevertheless, the proposal was to define a root type for time, where all time types derive.

The proposal was updated to reflect not only the advances in the Ada language (particularly in the Object-Oriented model), but also to simplify the model, where applications do not manage the delay queues, which are left to the runtime. There were still some open issues, such as the necessity for user-defined clocks to call back the runtime, for example to handle time discontinuities (jumping forward or back to the past), or the relationship between user-defined and calendar time.

There was some discussion in this issue, but no decision was taken at the workshop.

2.8 Generalising EDF support

The next proposal, also by Andy Wellings and Alan Burns [5], was to integrate into Ada the support to user-defined scheduling, leveraging on the fact that the Preemption Level Control Protocol introduced in Ada 2005 to support EDF can be used with several scheduling algorithms.

For this, a new mechanism would be required that permits to specify a task attribute on which to base dispatching decisions and to also control dispatching within a priority level. It would also be necessary to define new dispatching points, which would include the point where the value of a task's dispatching attribute was changed. Andy also presented some open issues, namely if other scheduling schemes other than EDF are useful, and if there are other resource sharing protocols that could be supported by Ada to allow for a wide range of scheduling schemes. There was some discussion on this issue, but no decision was taken in the workshop.

2.9 Non-Uniform Memory Access architectures

Andy Wellings presented the issue of supporting in Ada the NUMA (Non-Uniform Memory Access) multi-processor architectures. The accompanying paper [6] argues that the programming model should allow for a more visible mapping of the architecture at the programming level. Ada abstracts programmers away from the low-level architecture of the hardware, which although appropriate for SMP, does not permit to use NUMA architectures predictably and efficiently. There were a few comments on this issue, but it was not discussed further in the Workshop.

2.10 Execution-time accounting of interrupts handlers

Two independent proposals were submitted to the workshop on the issue of execution-time accounting of interrupts handlers. Both proposals noted that the current model is implementation defined but the usual approach is to charge the execution time of interrupts into the currently running task, which does not provide for accurate accounting. This is even more important as the introduction of timing events causes programmers to shift code from tasks to this low overhead mechanism, which is accounted as an interrupt.

However, the two approaches differed in the way the actual accounting was done.

In the proposal by Mario Aldea Rivas and Michael González-Harbour [7], the interrupts execution-time is accounted in a global "conceptual" task. An API is also provided for applications to monitor the time executed in interrupts. Therefore, applications not only have a more accurate measure of tasks' execution time but can also measure the time spent in interrupts.

The proposal by Kristoffer Gregertsen and Amund Skavhaug [8] defines a "pseudo" task for each interrupt priority, with execution time accounting done per priority.

After some discussion, consensus was formed that the separation of execution-time accounting was necessary, although that it would be more appropriate that accounting was performed per Interrupt_ID.

In a later round of discussion, an issue was raised that a simple solution would be to introduce an implementation advice that tasks should not be charged for the execution time of interrupts. Then, this would become a runtime quality issue. The proposal to introduce such implementation advice was approved with 16 votes in favour and 3 abstentions.

A second vote was made to decide if the workshop would propose a model, in case the implementation supported execution-time monitoring of interrupts. There were 9 votes in favour, 1 opposed and 11 abstentions. It was then decided that both Mario Aldea and Kristoffer Gregertsen would work on an API to be analyzed on the following day.

2.11 Ravenscar and multiprocessor issues

One of the issues which had been closed in the previous day was the integration of multiprocessor in the Ravenscar profile, and whether the profile should specify one particular multiprocessor scheduling model. There was a proposal by José Ruiz [9] for Ravenscar to determine that tasks would be statically allocated to processors, with a Ravenscar partition being a single scheduling domain as sanctioned in the previous session.

Doubts were raised however, particularly regarding whether task migration would be allowed, and whether the scheduling would be local to each processor or global to the domain. In particular, task migration under user control would allow more flexibility and efficiency. However, concerns were voiced that task migration could impact the certification of Ravenscar-based systems.

As for local versus global scheduling, it was argued that Ravenscar would need a local scheduling approach, with one ready queue per processor. This would impact the definition of scheduling domains agreed earlier, since in that definition, scheduling was global within the same domain.

Another issue with global scheduling and task migration was the impact on the locks of protected objects. In this model, all locks would have to be actual, and no longer priority based locks. The static allocation and local

scheduling approach would allow for more efficient implementations.

From the discussion, a list of possible models was introduced:

1. Tasks' fixed allocation, affinities specified by the user
2. Tasks' fixed allocation, affinities specified by the user, and allowing task migration
3. Tasks' fixed allocation, affinities specified by the runtime
4. Global scheduling, no fixed allocation

It was also considered that the model could be a mix, with some tasks fixed, while other were scheduled globally.

During the discussion, there were doubts as to whether all models were feasible, and whether the Ravenscar profile should specify a model at all. Multiprocessor scheduling is still fairly open area, with new models and algorithms every day, so maybe Ravenscar should be silent and afford implementations and programmers more flexibility. However, by doing that, different implementations could choose different models, something that the Ada standard tries to avoid.

Considering all of this, consensus was reached that the profile should not specify a model, but the workshop would recommend that Ravenscar is implemented together with option 1 above (tasks' fixed allocation, affinities specified by the user). This could be done through an implementation advice in the standard.

2.12 The Real-Time Framework

The Real-Time Framework had been proposed in the previous workshop, in 2008, and intended to provide a library of real-time utilities which could be used by application developers. In this workshop, Jorge Real proposed the integration of support for mode changes into the framework [10].

The proposal was based on the original implementation made available by Andy Wellings in 2008, which was also updated to work with a new version of the Ada compiler (at the time of the original framework, some Ada 2005 features were still not available in compilers). During the workshop Jorge also made available the updated framework code.

In the proposal, mode management is based on a synchronized interface, thus integrating the model with the object-oriented model of Ada. During the presentation of the mode manager, it was noted that in the implemented model, the mode manager depends on a user defined type (List_Of_Modes) and hence it is not independent from the application. It was then considered that a different model should be analysed.

Afterwards the discussion went on to consider whether the real-time utilities framework should be pursued for standardisation (for instance through a secondary standard) to make it more visible. A proposal was put forward to make it available in some form of collaborative platform, and announcing it to the Ada community as a work in progress.

It was decided to continue the work in an informal collaboration, and anyone interested in working on it to contact Jorge Real.

2.13 The Real-Time Transaction Model

The presentation of the real-time transaction model [11] was made by J. Javier Gutiérrez. The goal is to integrate distributed real-time transactions within the Distributed Systems Annex of Ada, allowing for a separation of concerns between the scheduling of both processing nodes and network, and the application code.

There was some discussion on whether an attempt should be made to standardize a real-time distributed model (or make it available as a technical report), such that all implementations would follow the same guidelines. The question was raised whether the integration of distribution and real-time would be useful and worth the effort. The issue has been around for several workshops, and it was generally agreed that this should be further pursued.

The discussion then went into a specific model for the Ravenscar profile. This particular model is not compatible with the profile, but it would be possible to make a compatible version. It was thus decided that a Ravenscar version should be proposed, with the intention to put forward a technical report. If the model was then accepted, it would be built also for full Ada.

3 Conclusions

The Language and Distribution session was mainly devoted to the discussion of the changes and additions to the language. An action list was permanently being built and updated reflecting the outcomes of the discussion, but for most of the issues actual decisions were deferred to the last, concluding, session of the workshop [12], where the final definition of the AIs to produce was completed.

References

[1] Rod White. Providing Additional Real-Time Capability and Flexibility for Ada 2005. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[2] Stephen Michell, Luke Wong, Brad Moore. Realtime Paradigms Needed Post Ada 2005. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[3] Luke Wong, Stephen Michell, Brad Moore. Named Memory Pool for Ada. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[4] Andy Wellings, Alan Burns. User-Defined Clocks. Is it the right time now? ACM Ada Letters, Volume XXX, Number 1, April 2010.

[5] Andy Wellings, Alan Burns. Generalizing the EDF Scheduling Support in Ada 2005. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[6] Andy Wellings, Abdul H. Malik, Neil Audsley, Alan Burns. Ada and cc-NUMA Architectures. What can

be achieved with Ada 2005? ACM Ada Letters, Volume XXX, Number 1, April 2010.

[7] Mario Aldea Rivas, Michael González Harbour. Execution time monitoring and interrupt handlers. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[8] Kristoffer Nyborg Gregertsen, Amund Skavhaug. Execution-time control for interrupt handling. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[9] José Ruiz. Towards a Ravenscar Extension for Multiprocessor Systems. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[10] Jorge Real and Alfons Crespo. Incorporating Operating Modes to an Ada Real-Time Framework. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[11] Héctor Pérez Tijero, J. Javier Gutiérrez, Michael González Harbour. Support for a real-time transactional model in distributed Ada. ACM Ada Letters, Volume XXX, Number 1, April 2010.

[12] Stephen Michell, Jorge Real. Conclusions of the 14th International Real-Time Ada Workshop. ACM Ada Letters, Volume XXX, Number 1, April 2010.

Conclusions of the 14th International Real-Time Ada Workshop

Chair: Stephen Michell
Rapporteur: Jorge Real

1 Introduction

The last session of IRTAW-14 was devoted to concluding on the results of the workshop, with the goal of prioritizing and selecting Ada Issues (AIs) to be produced and sent to the ISO/IEC JTC1/SC22/WG9 Ada Rapporteur Group (ARG). It allowed time for closing some open issues.

In this report, Sections 2 to 4 summarize the final discussion around some open issues. Section 5 reflects the list of AIs to be produced by the workshop. The plan for next meeting is considered in Section 6. There is a final consideration about concurrency vulnerabilities in Section 7. Finally, Section 8 concludes with the closing of IRTAW-14.

2 Barrier suspension objects

Stephen Michell summarized the proposal for Ada to include barrier suspension objects to allow parallel release of multiple readers upon a certain condition, expressed by means of a barrier [4]. This proposal is targeted to multiprocessor architectures and the goal is to allow true parallelism when multiple readers wait for data produced by a single writer task. In such cases, the maximum efficiency is achieved by broadcasting the data in parallel to all the interested reader tasks. This behaviour cannot be accomplished by means of an entry, as presently specified in Ada, due to the fact that only the first waiting task would be released upon barrier opening and then the barrier condition would need to be reevaluated every time a single task is served. The proposal can be supported in POSIX and in most present hardware implementations of a barrier.

It was noted that the definition of barrier suspension objects should be accompanied by pragma Preelaborate. The workshop did not see any need for requiring barrier suspension objects to be declared at library level. No implications were identified with respect to pragmas Intrinsic and Inline. There was unanimous support for the proposal.

3 Named storage pools

Named storage pools were proposed in [5]. They are motivated by the convenience to use storage pools specifically tied to one of the different kinds of memory available, since memory maps include different memory technologies in many systems (RAM, ROM, FLASH, etc.).

The workshop however did not find enough motivation for pushing for this change to the language: the proposal is not mature enough and there were opinions in the sense that there may be ways to achieve a similar functionality in current Ada. The proposal was therefore withdrawn. It was agreed, however, that there should be syntax added to Ada to permit the specification of an address for a declared storage pool.

4 Execution time control for interrupt handling

Kristoffer Gregertsen gave a summary of the proposal to introduce mechanisms for monitoring the execution time spent in servicing interrupts. This proposal was based on [3] and [1]. The proposed API (i) defines one execution-time clock per Interrupt ID, (ii) allows the mechanism to obtain the time spent in the handling of each interrupt, and (iii) also allows association of timers to those clocks.

Use of Ada.Interrupts.Interrupt_ID was preferred to using Task ID to identify execution time of the different interrupt handlers. A new package Ada.Execution Time.Interrupts contains the following subprogram:

```
function Clock (I: Ada.Interrupts.Interrupt_ID)
  return CPU_Time;
```

The function returns the execution time spent in handling the identified interrupt, or returns CPU Time First if the facility is not supported by the implementation.

The proposal was supported by 17 votes for, no vote against, and 3 abstentions. Hence an AI will be produced on this topic. Note that being a child package of Ada.Execution_Time, its implementation would be optional.

The workshop then considered a natural extension to this facility: timers for CPU time spent in interrupt handling. This feature would need the inclusion of interrupt clocks first, since timers rely on clocks. Although there was no objection to the interface described in [3], there was no general support for pushing this feature forward to standardization (4 votes for, 2 against and 13 abstentions). The workshop however agreed to suggest to ARG the inclusion of an implementation advice stating that this service, if implemented, should stick to the proposed interface:

```
with Ada.Interrupts;
package Ada.Execution_Time.Timers.Interrupts is
  type Timer (I: Ada.Interrupts.Interrupt_ID)
    is new Ada.Execution_Time.Timers.Timer(
          Ada.Task_Identification.Null_Task_Id'Access)
    with private;
```

This paper previously appeared in ACM Ada Letters, Volume XXX, Number 1, April 2010; reprinted with permission

```
private
. . .
end Ada.Execution_Time.Timers.Interrupts;
```

5 Wrapping up

Alan Burns prepared the list of topics about which the workshop agreed to produce new AIs. The list was reviewed and the different items were assigned to those in charge of writing them. The final list considers:

1. Addition of affinity support packages, interrupt affinities and considerations about spin locking — A. Burns and A. Wellings.
2. Change definition of group budgets to include processor, with default to processor 1 — A. Burns and A. Wellings.
3. Addition of an implementation advice to allow for multiprocessor execution of Ravenscar programs — J. Ruiz.
4. Addition of timers to the Ravenscar profile (a maximum of one timer per task) — T. Vardanega. [1]
5. Add the definition of barrier suspension objects — S. Michell.
6. Implementation advice on interrupt monitoring — M. Gonz´alez and M. Aldea.
7. Addition of operations yield and yield to higher priority in non-preemptive scheduling — A. Burns.
8. Deadlines in synchronous task control — A. Burns.
9. Addition of interrupt execution-time accounting clocks — M. González.

6 Conclusion and next IRTAW

Deadlines were set for finalization of session reports, production of final versions of the position papers, and writing of the AIs to be sent to ARG. Alan Burns will centralize the AIs and propose them in the next ARG meeting.

There was general agreement about the need of future editions of IRTAW. The next edition will be organized by Michael González in the Santander area, in Spain. The workshop is scheduled for April or May 2011, hence we leave some 18 months between editions 14 and 15. Mario Aldea will head the role of Program Committee Chair.

[1] Tullio will check that the Ravenscar model is not broken with this addition and get feedback from implementors — perhaps add a restriction (e.g. Max_Nr_Of_Timers_Per_Task and set it to 1 for Ravenscar).

7 Consideration of concurrency vulnerabilities

The ISO/IEC JTC 1/SC 22/WG 23 (WG23, for short) is preparing a technical report about Programming Language Vulnerabilities. One of the items in the workshop agenda was to note the absence of concurrency-related vulnerabilities in the technical report being prepared, as reflected in the position paper [2].

The workshop decided to submit this position paper to WG23, after receiving comments and suggestions for improvement from participants at the workshop. Miguel Pinho noted that multiprocessor execution may be yet another source of vulnerabilities worth considering.

8 Closing

There being no other pending issues, Stephen Michell closed the session and the workshop. The workshop thanked specially the presence of first-time participants and encouraged them to continue to do so. All thanked Tullio Vardanega for the splendid local arrangement.

References

[1] M. Aldea and Michael González-Harbour. Execution time monitoring and interrupt handlers. Ada Letters, ACM Ada Letters, Volume XXX, Number 1, April 2010.

[2] A. Burns and A. Wellings. Language vulnerabilities - let's not forget concurrency. Ada Letters, ACM Ada Letters, Volume XXX, Number 1, April 2010.

[3] K. Gregertsen and A. Skavhaug. Execution-time control for interrupt handling. Ada Letters, ACM Ada Letters, Volume XXX, Number 1, April 2010.

[4] S. Michell, L. Wong, and B. Moore. Real-Time paradigms needed post Ada 2005. Ada Letters, ACM Ada Letters, Volume XXX, Number 1, April 2010.

[5] L. Wong, S. Michell, and B. Moore. Named memory pool for Ada. Ada Letters, ACM Ada Letters, Volume XXX, Number 1, April 2010.

Progress Report from the 14th International Real-Time Ada Workshop – IRTAW14

Alan Burns
Department of Computer Science, University of York, UK

Abstract

This paper reviews the outcomes of the latest workshop in the IRTAW series. Specifically it looks at the impact the workshop is having on the current effort to define the 2012 amendment to Ada.

Keywords: real-time, Ada.

1 Introduction

The 14th International Real-Time Ada workshop took place in Portovenere, Italy during 7-9 October 2009. A total of fourteen papers were accepted for the workshop. But in keeping with the tradition of the IRTAW series these papers were not formally presented but were made available to the 24 delegates before the event – they formed the background to the discussions that took place. The papers themselves have been published by Ada Letters in Volume XXX, Number 1 (April 2010); a list of the papers is included in Appendix A.

The workshop was organised into a number of discussion sessions:

- Multiprocessor Systems,
- Language and Distribution Issues, and
- Conclusions and Recommendations.

Each of these sessions produced a summary report that were also published in the above referenced volume of Ada Letters [4].

The objectives of the workshop include a wish to consider all relevant language issues to do with the support of real-time applications. This includes experiences in implementing and using current language features, and the exploration of possible new features. Some of these features are 'in the long term', but others are relevant to the current effort to define the 2012 amendment to Ada. In the remainder of this paper, we focus on this latter objective, and review the progress that is currently been made to influence the development of Ada 2012.

2 Recommendations from IRTAW14

The discussions during the workshop produced the following recommendations for consideration by the Ada 2012 revision process:

1. To support multiprocessor-based systems the notion of an allocation domain was developed. The key point about an allocation domain is that it defines a set of CPUs on which tasks are globally scheduled. There is a default allocation domain (the System domain) that is the set of all processors allocated to a program. The assumption is that this is a fixed set of processors that does not change during the execution of the program. The default behaviour of an Ada program if it does not specify anything is therefore global scheduling of all tasks across all processors. But tasks with an allocation domain can be assigned a specific CPU to support fully partitioned allocations.
2. A static Ravenscar specific solution to the allocation problem for multicore targets was defined for inclusion in the profile's definition.
3. Group Timers should be confined to just a single processor – to manage parallel use of budgets was deemed too problematic.
4. Where possible the time spent in interrupt handlers should not be added to a task's execution time clock. Ideally the time spent handling interrupts should be available via a 'special' interrupt clock.
5. For non-preemtpive scheduling it is useful to distinguish between yielding to strictly higher priority tasks, and yielding to equal or higher priority. The use of an explicit yield procedure would also help the readability of programs (the current alternative is to make calls of delay 0.0, or delay until <some time in the past>).
6. For EDF scheduled sporadic tasks, where a task's release is controlled by a synchronous task control object, there is a need to be able to suspend with one deadline but to have another deadline when next released. This is equivalent to the delay until and set deadline primitive for controlling EDF scheduled periodic tasks. A 'Suspend_Until_True_And_Set_Deadline' procedure was proposed.
7. For parallel hardware, where data-oriented parallelism is being employed, a thread barrier is often supported. This allows a set of threads to be blocked until the final thread (of the set) arrives. All threads are then released (with one of the threads being identified as having a special status so that is can manipulate the barrier). A task-based primitive for Ada is proposed.

[4] Editor's note: the session summaries are also available in this issue of the Ada User Journal.

8. Another addition to the Ravenscar profile was to allow Timers (perhaps a maximum of one per task) to be included in the profile.

Of course there were a number of other issues and topics discussed that may lead to changes to Ada in the future. Among this list are: Ravenscar and EDF scheduling, named memory pools, barriers functions in protected objects, user-defined clocks, support for NUMA architectures, the real-time framework (set of utilities) and further support for distributed applications. These, and other, topics will be considered at the next IRTAW event (in September 2011).

3 AIs in the Ada 2012 process

The definition and maintenance of the Ada language is the responsibility of the ARG, a working committee of WG9, itself an ISO/IEC committee. The ARG manages its work by placing all possible language changes into an AI (Ada Issue). For Ada 2012 there are over 200 such AIs. From the above list of topics that the workshop defined as being of relevance to Ada 2012 the following AIs were developed (the number in brackets refers to the above numbered list):

AI-166: Yield for non-preemptive dispatching (5).

AI-167: Managing affinities for programs executing on multiprocessors (1).

AI-168: Extended suspension objects (6).

AI-169: Defining group budgets for multiprocessor platforms (3).

AI-170: Monitoring the time spent in Interrupt Handlers, and providing a clock to read these values (4).

AI-171: Pragma CPU and Ravenscar Profile (1,2).

AI-172: Extension to Ravenscar Profile (8).

AI-174: Implement Task barriers in Ada (7).

AI-210: Correct Timing_Events metric.

AI-211: No_Relative_Delay should not allow relative timing events.

The latter two very minor issues were clear errors in the 2005 definition of Ada and must therefore be corrected.

Other AIs that have relevance to the real-time community, though not arising from this workshop are:

AI-30: Requeue on synchronized interfaces (came from last IRTAW, allows for more general patterns to be developed),

AI-94: Timing_Events should not require deadlock (a common programming idiom is to set a handler while executing a handler – obviously this should not lead to deadlock!),

AI-117: Memory barriers and Volatile objects (a class of non-locking algorithm for parallel hardware require that assignments to shared variables are not reordered – Ada's definition of Volatile needs to ensure this),

AI-119: Package Calendar, Daylight Savings Time, and UTC_Offset,

AI-202: Task_Termination and Exceptions raised during finalization.

All of these AI can be obtained from the Ada Conformity Assessment Authority home page: www.ada-auth.org/.

4 Progress of the real-time AIs

All but one of the 'workshop' AIs is currently making progress through the ARG procedures. The one that has been dropped is AI-172. It was felt that the Ravenscar profile was a significant 'brand' for Ada, and that changes to it should not be made likely. Further consideration of the increase in run-time complexity was needed. This is likely to be taken up at the next IRTAW.

Of the other AIs, many have already been 'concluded' and have either been progressed through the pipeline to WG9, or are awaiting the final word-smiting. The ones currently been worked on are those concerned with multiprocessor scheduling (AI-167 and AI-171). These represent the more significant changes and hence it is not surprising that they still require further work. However, it is still the view of ARG that these should make it through to Ada 2012.

The process by which the ARG transforms ideas presented to it by the IRTAW inevitable leads to many necessary changes. Often the final language feature is quite different from what was discussed at the workshop. Nevertheless, the essential need always remains at the heart of the discussions and the final amendment does indeed address the issue raised.

To give an example of this process, consider AI-166. The initial recommendation from the workshop is that the following package be added to the Standard:

```
package Ada.Dispatching.Non_Preemptive is
  procedure Yield_To_Higher;
    procedure Yield; - - Bounded error if
     - - executed within a protected operation
  end Ada.Dispatching.Non_Preemptive;
```

After seven iteration of this definition the final language change recommendation is to add to package `Ada.Dispatching` the following procedure:

```
procedure Yield; -- Bounded error if
- - executed within a protected operation
```

and to include the following new package:

```
package Ada.Dispatching.Non_Preemptive is
  pragma Preelaborate(Non_Preemptive);
  procedure Yield_To_Higher;
  procedure Yield_To_Same_Or_Higher
    renames Yield;
end Ada.Dispatching.Non_Preemptive;
```

Hence the functionality is split between two packages and the Yield procedure becomes available even if preemptive dispatching is being used.

To give an example of a complete set of language changes concerned with just a single well-focused issue, consider AI-168. The required wording change from this AI is:

Add after D.10(5):

The following language-defined library package exists:

```
with Ada.Real_Time;
package Ada.Synchronous_Task_Control.EDF is
    procedure
    Suspend_Until_True_And_Set_Deadline
    (S : in out Suspension_Object;
     TS : in Ada.Real_Time.Time_Span);
end Ada.Synchronous_Task_Control.EDF;
```

Add after D.10(10):

The procedure Suspend_Until_True_And_Set_Deadline blocks the calling task until the state of the object S is True; at that point the task becomes ready with a deadline of Ada.Real_Time.Clock + TS, and the state of the object becomes False. Suspend_Until_True_And_Set_Deadline is a potentially blocking operation.

Add after D.10(11):

NOTE: More complex schemes, such as setting the deadline relative to when Set_True is called, can be programmed using a protected object.

The other major changes to the workshop's proposals concern the support for affinities for programs running on multicore or multiprocessor platforms. Here the terminology has changed (from *allocation domains* to *dispatching domains*) and the functionality has been reduced (no longer will each domain be able to specify different scheduling rules). But again the essential requirements identified by the workshop are being met.

5 Conclusions

Just as the previous thirteen workshops in the IRTAW series have influenced the continuing development of Ada, the 14th event proved to again generate ideas that keep Ada at the forefront of languages in terms of its support for real-time programming. Many aspects of embedded and real-time systems are having to face up to the challenges that new parallel hardware is generating. There is much to be done in this area, but Ada has made a start by introducing the notion of affinity into the set of abstractions that it makes available to programmers.

It is most likely that future workshops will continue to focus on this crucial area. But it is also important that implementations become available that allow these new features to be used. Only though experience will the new abstractions be tested and evaluated as to whether they are fit for purpose. If they are, we can continue to progress the Ada language, if they are not then alternatives must be developed. In both of these endeavours the IRTAW series will have a role.

The 15th IRTAW will take place in Spain in mid September 2011. A call for papers will be available shortly; readers of the Ada User Journal are encouraged to consider participating.

Appendix A

The following papers were accepted for the Workshop and are now available via Volume XXX, Number 1 (April 2010) of Ada Letters:

[1] Supporting Execution on Multiprocessor Platforms - A. Burns and A.J. Wellings.

[2] Language Vulnerabilities - Let's not forget Concurrency - A. Burns and A.J. Wellings.

[3] Execution-time control for interrupt handling - Kristoffer Nyborg Gregertsen and Amund Skavhaug.

[4] Temporal Isolation with the Ravenscar Profile and Ada 2005 - Enrico Mezzetti, Marco Panunzio and Tullio Vardanega

[5] Named Memory Pool for Ada - Luke Wong, Stephen Michell and Brad Moore.

[6] Realtime Paradigms Needed Post Ada 2005 – Stephen Michell, Luke Wong and Brad Moore.

[7] Execution time monitoring and interrupt handlers, Position Statement – Mario Aldea Rivas and Michael Gonzales Harbour.

[8] Incorporating Operating Modes to an Ada Real-Time Framework - Jorge Real and Alfons Crespo.

[9] Towards a Ravenscar Extension for Multi-Processor Systems - Jose F. Ruiz.

[10] Support for a real-time transactional model in distributed Ada - Hector Perez Tijero, Javier Gutierrez and Michael G. Harbour.

[11] User-Defined Clocks. Is it the right time now? - A.J. Wellings and A. Burns.

[12] Generalizing the EDF Scheduling Support in Ada 2005 - A.J. Wellings and A. Burns.

[13] Ada and CC-NUMA Architectures: What can be achieved with Ada 2005? - A.J. Wellings, A.H. Malik, N.C. Audsley and A. Burns.

[14] Providing Additional Real-Time Capability and Flexibility for Ada 2005 – Rod White.

Ada and the Software Vulnerabilities Project: the SPARK Annex

Alan Burns, FREng (ed.)

Department of Computer Science, University of York, York YO1 5DD UK; Tel: +44 (0)1904 432779; email: burns@cs.york.ac.uk

Joyce L. Tokar, PhD (ed.)

Pyrrhus Software, PO Box 1352, Phoenix, AZ, 85001-1352, USA.; Tel: +1 602373 0713; email: tokar@pyrrhusoft.com

Stephen Baird, John Barnes, Rod Chapman, Gary Dismukes, Michael González-Harbour, Stephen Michell, Brad Moore, Luís Miguel Pinho, Erhard Ploedereder, Jorge Real, J.P. Rosen, Ed Schonberg, S. Tucker Taft, T. Vardanega

Abstract

*In a previous article [1] we published the Ada [2] Annex to the Technical Report (TR) on software vulnerabilities [3], developed by ISO/IEC JTC 1/SC 22/WG 23. This article completes this work, with the annex concerning SPARK [4] *.*

Keywords: software vulnerabilities, software vulnerability, Ada, SPARK.

1 Introduction

Software vulnerabilities are defined as a property of a system security, requirements, design, implementation, or operation that could be accidentally triggered or intentionally exploited and result in a security failure [5]. Work on software vulnerabilities and how they enable software applications to be infiltrated and corrupted continues to be of interest world. Working Group 23 (WG 23) of the Programming Languages Subcommittee (SC 22) of the International Organization of Standards (ISO) has recently completed a Technical Report that identifies and enumerates a collection of software vulnerabilities in existing programming languages [3]. Annexes to this document are being developed to identify if the vulnerabilities defined in the TR exist in various programming languages.

A workshop was conducted in parallel with the 14th International Conference on Reliable Software Technologies – Ada-Europe 2009 to initiate the development of content of an Annex to the Technical Report that documents its applicability to the Ada and SPARK programming languages. The results of this workshop were published in [6]. Another workshop was conducted in parallel with the 2009 SIGAda conference. Work continued on this document over the course of 2009 and was completed in a short workshop at the 15th International Conference on Reliable Software Technologies – Ada-Europe 2010. A previous article [1] published the final draft copy of the Ada Annex to the WG 23 TR submitted to WG 23 for inclusion in the TR. This article completes the work, providing the SPARK annex developed by Altran-Praxis.

Note, within the WG 23 TR each vulnerability is assigned a unique identifier such as RIP for the Inheritance vulnerability. Since the WG 23 TR was under development during the work on this Annex and there is an expectation that more vulnerabilities will be added to the TR, the sections in the Ada and SPARK annexes include their corresponding unique identifier in the section heading.

* For completeness, the article republishes and adapts the Introduction section of [1].

References

[1] Burns, A., Tokar, J. L. (Eds.), Ada and the Software Vulnerabilities Project, in Ada User Journal, Vol. 31, number 3, September 2010, pp. 191-215.

[2] Taft, S. Tucker, Duff, R. A., Brukardt, R. L., Ploedereder, E., Leroy, P, Ada Reference Manual, LNCS 4348, Springer, Heidelberg, 2006.

[3] ISO/IEC JTC 1/SC 22 N 4522, ISO/IEC TR 24772, Information Technology — Programming Languages — Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use, 7 November 2009.

[4] SPARK Language Definition: "SPARK95: The SPADE Ada Kernel (Including RavenSPARK)" Available at www.altran-praxis.com.

[5] NIST Special Publication 268, "Source Code Security Analysis Tool Functional Specification Version 1.0," May 2007.

[6] Proceedings of the Software Vulnerabilities Workshop of Ada-Europe 2009, in Ada User Journal, Volume 30, Number 3, September 2009, pp. 174-192.

Annex SPARK – Final Draft

SPARK.Specific information for vulnerabilities

SPARK.1 Identification of standards and associated documentation

See Ada.1 *, plus the references below. In the body of this annex, the following documents are referenced using the short abbreviation that introduces each document, optionally followed by a specific section number. For example "[SLRM 5.2]" refers to section 5.2 of the SPARK Language Definition.

[SLRM] SPARK Language Definition: "SPARK95: The SPADE Ada Kernel (Including RavenSPARK)" Latest version always available from www.altran-praxis.com.

[SB] "High Integrity Software: The SPARK Approach to Safety and Security." John Barnes. Addison-Wesley, 2003. ISBN 0-321-13616-0.

[IFA] "Information-Flow and Data-Flow Analysis of while-Programs." Bernard Carré and Jean-Francois Bergeretti, ACM Transactions on Programming Languages and Systems (TOPLAS) Vol. 7 No. 1, January 1985. pp 37-61.

[LSP] "A behavioral notion of subtyping." Barbara Liskov and Jeannette Wing. ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 16, Issue 6 (November 1994), pp. 1811 - 1841.

SPARK.2 General terminology and concepts

The SPARK language is a contractualized subset of Ada, specifically designed for high-assurance systems. SPARK is designed to be amenable to various forms of static analysis that prevent or mitigate the vulnerabilities described in this TR.

This section introduces concepts and terminology which are specific to SPARK and/or relate to the use of static analysis tools.

Soundness
This concept relates to the absence of false-negative results from a static analysis tool. A false negative is when a tool is posed the question "Does this program exhibit vulnerability X?" but incorrectly responds "no." Such a tool is said to be **unsound** for vulnerability X. A sound tool effectively finds **all** the vulnerabilities of a particular class, whereas an unsound tool only finds some of them.

The provision of soundness in static analysis is problematic, mainly owing to the presence of unspecified and undefined features in programming languages. Claims of soundness made by tool vendors should be carefully evaluated to verify that they are reasonable for a particular language, compilers and target machines. Soundness claims are always underpinned by assumptions (for example, regarding the reliability of memory, the correctness of compiled code and so on) that should also be validated by users for their appropriateness.

Static analysis techniques can also be **sound in theory** – where the mathematical model for the language semantics and analysis techniques have been formally stated, proved, and reviewed – but **unsound in practice** owing to defects in the implementation of analysis tools. Again, users should seek evidence to support any soundness claim made by language designers and tool vendors. A language which is **unsound in theory** can never be sound in practice.

The single overriding design goal of SPARK is the provision of a static analysis framework which is **sound in theory,** and as **sound in practice** as is reasonably possible.

In the subsections below, we say that SPARK **prevents** a vulnerability if supported by a form of static analysis which is sound in theory. Otherwise, we say that SPARK **mitigates** a particular vulnerability.

SPARK Processor
We define a "SPARK Processor" to be a tool that implements the various forms of static analysis required by the SPARK language definition. Without a SPARK Processor, a program cannot reasonably be claimed to be SPARK at all, much in the same way as a compiler checks the static semantic rules of a standard programming language.

In SPARK, certain forms of analysis are said to be **mandatory** – they are required to be implemented and programs must pass these checks to be valid SPARK. Examples of mandatory analyses are the enforcement of the SPARK language subset, static semantic analysis (e.g. enhanced type checking) and information flow analysis [IFA].

Some analyses are said to be **optional** – a user may choose to enable these additional analyses at their discretion. The most notable example of an optional analysis in SPARK is the generation of verification conditions and their proof using a theorem proving tool. Optional analyses may provide greater depth of analysis, protection from additional vulnerabilities, and so on, at the cost of greater analysis time and effort.

Failure modes for static analysis
Unlike a language compiler, a user can always choose not to, or might just forget to run a static analysis tool. Therefore, there are two modes of failure that apply to all vulnerabilities:

* Editor's note: The Ada Annex is published in the September 2010 issue of the Ada User Journal (Vol. 31, n. 3).

1. The user fails to apply the appropriate static analysis tool to their code.
2. The user fails to review or mis-interprets the output of static analysis.

SPARK.3.BRS Obscure Language Features [BRS]

SPARK mitigates this vulnerability.

SPARK.3.BRS.1 Terminology and features

As in Ada.3.BRS.1.

SPARK.3.BRS.2 Description of vulnerability

As in Ada.3.BRS.2.

SPARK.3.BRS.3 Avoiding the vulnerability or mitigating its effects

The design of the SPARK subset avoids many language features that might be said to be "obscure" or "hard to understand", such as controlled types, unrestricted tasking, anonymous access types and so on.

SPARK goes further, though, in aiming for a completely *unambiguous* semantics, removing all erroneous and implementation-dependent features from the language. This means that a SPARK program should have a single meaning to programmers, reviewers, maintainers and all compilers.

SPARK also bans the aliasing, overloading, and redeclaration of names, so that one entity only ever has one name and one name can denote at most one entity, further reducing the risk of mis-understanding or mis-interpretation of a program by a person, compiler or other tools.

SPARK.3.BRS.4 Implications for standardization

None.

SPARK.3.BRS.5 Bibliography

None.

SPARK.3.BQF Unspecified Behaviour [BQF]

SPARK prevents this vulnerability.

SPARK.3.BQF.1 Terminology and features

As in Ada.3.BQF.1.

SPARK.3.BQF.2 Description of vulnerability

As in Ada.3.BQF.2.

SPARK.3.BQF.3 Avoiding the vulnerability or mitigating its effects

SPARK is designed to eliminate all unspecified language features and bounded errors, either by subsetting to make the offending language feature illegal in SPARK, or by ensuring that the language has neutral semantics with regard to an unspecified behaviour.

"Neutral semantics" means that the program has identical meaning regardless of the choice made by a compiler for a particular unspecified language feature.

For example:

- Unspecified behaviour as a result of parameter-passing mechanism is avoided through subsetting (no access types) and analysis to make sure that formal and global parameters do not overlap and create a potential for aliasing [SLRM 6.4].

- Dependence on evaluation order is prevented through analysis so that all expressions in SPARK are free of side-effects and potential run-time errors. Therefore, any evaluation order is allowed and the result of the evaluation is the same in all cases [SLRM 6.1].

- Bounded error as a result of uninitialized variables is prevented by application of static information flow analysis [IFA].

SPARK.3.BQF.4 Implications for standardization

None.

SPARK.3.BQF.5 Bibliography

None.

SPARK.3.EWF Undefined Behaviour [EWF]

SPARK prevents this vulnerability.

SPARK.3.EWF.1 Terminology and features

As in Ada.3.EWF.1.

SPARK.3.EWF.2 Description of vulnerability

As in Ada.3.EWF.2.

SPARK.3.EWF.3 Avoiding the vulnerability or mitigating its effects

SPARK prevents all erroneous behaviour, either through subsetting or static analysis [SB 1.3].

SPARK.3.EWF.4 Implications for standardization

None.

SPARK.3.EWF.5 Bibliography

None.

SPARK.3.FAB Implementation-Defined Behaviour [FAB]

SPARK mitigates this vulnerability.

SPARK.3.FAB.1 Terminology and features

As in Ada.3.FAB.1.

SPARK.3.FAB.2 Description of vulnerability

As in Ada.3.FAB.2.

SPARK.3.FAB.3 Avoiding the vulnerability or mitigating its effects

SPARK allows a number of implementation-defined features as in Ada. These include:

- The range of predefined integer types.
- The range and precision of predefined floating-point types.
- The range of System.Any_Priority and its subtypes.
- The value of constants such as System.Max_Int, System.Min_Int and so on.
- The selection of T'Base for a user-defined integer or floating-point type T.
- The rounding mode of floating-point types.

In the first four cases, static analysis tools can be configured to "know" the appropriate values [SB 9.6]. Care must be taken to ensure that these values are correct for the intended implementation. In the fifth case, SPARK defines a contract to indicate the choice of base-type, which can be checked by a pragma Assert. In the final case, additional static analysis of numerical precision must be performed by the user to ensure the correctness of floating-point algorithms.

SPARK.3.FAB.4 Implications for standardization

None.

SPARK.3.FAB.5 Bibliography

None.

SPARK.3.MEM Deprecated Language Features [MEM]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.MEM.

SPARK.3.NMP Pre-Processor Directives [NMP]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.NMP.

SPARK.3.NAI Choice of Clear Names [NAI]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.NAI.

SPARK.3.AJN Choice of Filenames and other External Identifiers [AJN]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.AJN.

SPARK.3.XYR Unused Variable [XYR]

SPARK mitigates this vulnerability.

SPARK.3.XYR.1 Terminology and features

As in Ada.3.XYR.1.

SPARK.3.XYR.2 Description of vulnerability

As in Ada.3.XYR.2.

SPARK.3.XYR.3 Avoiding the vulnerability or mitigating its effects

As in Ada.3.XYR.3. Also, SPARK is designed to permit sound static analysis of the following cases [IFA]:

- Variables which are declared but not used at all.
- Variables which are assigned to, but the resulting value is not used in any way that affects an output of the enclosing subprogram. This is called an "ineffective assignment" in SPARK.

SPARK.3.XYR.4 Implications for standardization

None.

SPARK.3.XYR.5 Bibliography

None.

SPARK.3.YOW Identifier Name Reuse [YOW]

SPARK prevents this vulnerability.

SPARK.3.YOW.1 Terminology and features

As in Ada.3.YOW.1.

SPARK.3.YOW.2 Description of vulnerability

As in Ada.3.YOW.2.

SPARK.3.YOW.3 Avoiding the vulnerability or mitigating its effects

This vulnerability is prevented through language rules enforced by static analysis. SPARK does not permit names in local scopes to redeclare and hide names that are already visible in outer scopes [SLRM 6.1].

SPARK.3.YOW.4 Implications for standardization

None.

SPARK.3.YOW.5 Bibliography

None.

SPARK.3.BKL Namespace Issues [BJL]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.BJL.

SPARK.3.IHN Type System [IHN]

SPARK mitigates this vulnerability.

SPARK.3.IHN.1 Terminology and features

SPARK's type system is a simplification of that of Ada. Both Explicit and Implicit conversions are permitted in SPARK, as is instantiation and use of Unchecked_Conversion [SB 1.3].

A design goal of SPARK is the provision of *static type safety,* meaning that programs can be shown to be free from all run-time type failures using entirely static analysis. If this optional analysis is achieved, a SPARK program should never raise an exception at run-time.

SPARK.3.IHN.2 Description of vulnerability

As in Ada.3.IHN.2 for Unchecked_Conversion.

SPARK.3.IHN.3 Avoiding the vulnerability or mitigating its effects

Vulnerabilities relating to value conversions, exceptions, and assignments are mitigated by static analysis. Vulnerabilities relating to the use of Unchecked_Conversion are as in Ada.

SPARK.3.IHN.4 Implications for standardization

None.

SPARK.3.IHN.5 Bibliography

None.

SPARK.3.STR Bit Representation [STR]

SPARK mitigates this vulnerability.

SPARK.3.STR.1 Terminology and features

As in Ada.3.STR.1.

SPARK.3.STR.2 Description of vulnerability

SPARK is designed to offer a semantics which is independent of the underlying representation chosen by a compiler for a particular target machine. Representation clauses are permitted, but these do not affect the semantics as seen by a static analysis tool [SB 1.3].

SPARK.3.STR.3 Avoiding the vulnerability or mitigating its effects

As in Ada.3.STR.4.

SPARK.3.STR.4 Implications for standardization

None.

SPARK.3.STR.5 Bibliography

None.

SPARK.3.PLF Floating-point Arithmetic [PLF]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.PLF.

SPARK.3.CCB Enumerator Issues [CCB]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.CCB.

SPARK.3.FLC Numeric Conversion Errors [FLC]

SPARK prevents this vulnerability.

SPARK.3.FLC.1 Terminology and features

As in Ada.3.FLC.1.

SPARK.3.FLC.2 Description of vulnerability

As in Ada.3.FLC.2.

SPARK.3.FLC.3 Avoiding the vulnerability or mitigating its effects

SPARK is designed to be amenable to static verification of the absence of predefined exceptions, and in particular all cases covered by this vulnerability [SB 11]. All numeric conversions (both explicit and implicit) give rise to a verification condition that must be discharged, typically using an automated theorem-prover.

SPARK.3.FLC.4 Implications for standardization

None.

SPARK.3.FLC.5 Bibliography

None.

SPARK.3.CJM String Termination [CJM]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.CJM.

SPARK.3.XYX Boundary Beginning Violation [XYX]

SPARK prevents this vulnerability.

SPARK.3.XYX.1 Terminology and features

As in Ada.3.XYX.1.

SPARK.3.XYX.2 Description of vulnerability

As in Ada.3.XYX.2.

SPARK.3.XYX.3 Avoiding the vulnerability or mitigating its effects

SPARK is designed to permit static analysis for all such boundary violations, through techniques such as theorem proving or abstract interpretation [SB 11].

SPARK programs that have been subject to this level of analysis can be compiled with run-time checks suppressed, supported by a body of evidence that such checks could never fail, and thus removing the possibility of erroneous execution.

SPARK.3.XYX.4 Implications for standardization

None.

SPARK.3.XYX.5 Bibliography

None.

SPARK.3.XYZ Unchecked Array Indexing [XYZ]

SPARK prevents this vulnerability.

SPARK.3.XYZ.1 Terminology and features

As in Ada.3.XYZ.1.

SPARK.3.XYZ.2 Description of vulnerability

As in Ada.3.XYZ.2.

SPARK.3.XYZ.3 Avoiding the vulnerability or mitigating its effects

As per SPARK.3.XYX.3 – this vulnerability is eliminated in SPARK by static analysis using the same techniques.

SPARK.3.XYZ.4 Implications for standardization

None.

SPARK.3.XYZ.5 Bibliography

None.

SPARK.3.XYW Unchecked Array Copying [XYW]

SPARK prevents this vulnerability.

SPARK.3.XYW.1 Terminology and features

As in Ada.3.XYW.1.

SPARK.3.XYW.2 Description of vulnerability

As in Ada.3.XYW.2.

SPARK.3.XYW.3 Avoiding the vulnerability or mitigating its effects

Array assignments in SPARK are only permitted between objects that have statically matching bounds, so there is no

possibility of an exception being raised [SB 5.5, SLRM 4.1.2]. Ada's "slicing" and "sliding" of arrays is not permitted in SPARK, so this vulnerability cannot occur.

SPARK.3.XYW.4 Implications for standardization

None.

SPARK.3.XYW.5 Bibliography

None.

SPARK.3.XZB Buffer Overflow [XZB]

SPARK prevents this vulnerability.

SPARK.3.XZB.1 Terminology and features

As in Ada.3.HCF.1.

SPARK.3.XZB.2 Description of vulnerability

As in Ada.3.XZB.2.

SPARK.3.XZB.3 Avoiding the vulnerability or mitigating its effects

As per SPARK.3.XYX.3 – this vulnerability is eliminated in SPARK by static analysis using the same techniques.

SPARK.3.XZB.4 Implications for standardization

None.

SPARK.3.XZB.5 Bibliography

None.

SPARK.3.HCF Pointer Casting and Pointer Type Changes [HCF]

SPARK prevents this vulnerability.

SPARK.3.HCF.1 Terminology and features

As in Ada.3.HCF.1.

SPARK.3.HCF.2 Description of vulnerability

As in Ada.3.HCF.2.

SPARK.3.HCF.3 Avoiding the vulnerability or mitigating its effects

This vulnerability cannot occur in SPARK, since the SPARK subset forbids the declaration or use of access (pointer) types [SB 1.3, SLRM 3.10].

SPARK.3.HCF.4 Implications for standardization

None.

SPARK.3.HCF.5 Bibliography

None.

SPARK.3.RVG Pointer Arithmetic [RVG]

SPARK prevents this vulnerability.

SPARK.3.RVG.1 Terminology and features

As in Ada.3.RVG.1.

SPARK.3.RVG.2 Description of vulnerability

As in Ada.3.RVG.2.

SPARK.3.RVG.3 Avoiding the vulnerability or mitigating its effects

This vulnerability cannot occur in SPARK, since the SPARK subset forbids the declaration or use of access (pointer) types [SLRM 3.10].

SPARK.3.RVG.4 Implications for standardization

None.

SPARK.3.RVG.5 Bibliography

None.

SPARK.3.XYH Null Pointer Dereference [XYH]

SPARK prevents this vulnerability.

SPARK.3.XYH.1 Terminology and features

As in Ada.3.XYH.1.

SPARK.3.XYH.2 Description of vulnerability

As in Ada.3.XYH.2.

SPARK.3.XYH.3 Avoiding the vulnerability or mitigating its effects

This vulnerability cannot occur in SPARK, since the SPARK subset forbids the declaration or use of access (pointer) types [SLRM 3.10].

SPARK.3.XYH.4 Implications for standardization

None.

SPARK.3.XYH.5 Bibliography

None.

SPARK.3.XYK Dangling Reference to Heap [XYK]

SPARK prevents this vulnerability.

SPARK.3.XYK.1 Terminology and features

As in Ada.3.XYK.1.

SPARK.3.XYK.2 Description of vulnerability

As in Ada.3.XYK.2.

SPARK.3.XYK.3 Avoiding the vulnerability or mitigating its effects

This vulnerability cannot occur in SPARK, since the SPARK subset forbids the declaration or use of access (pointer) types [SLRM 3.10].

SPARK.3.XYK.4 Implications for standardization

None.

SPARK.3.XYK.5 Bibliography

None.

SPARK.3.SYM Templates and Generics [SYM]

At the time of writing, SPARK does not permit the use of generics units, so this vulnerability is currently prevented. In future, the SPARK language may be extended to permit generic units, in which case section Ada.3.SYM applies.

SPARK.3.RIP Inheritance [RIP]

SPARK mitigates this vulnerability.

SPARK.3.RIP.1 Terminology and features

As in Ada.3.RIP.1.

SPARK.3.RIP.2 Description of vulnerability

As in Ada.3.RIP.1.

SPARK.3.RIP.3 Avoiding the vulnerability or mitigating its effects

SPARK permits only a subset of Ada's inheritance facilities to be used. Multiple inheritance, class-wide operations and dynamic dispatching are not permitted, so all vulnerabilities relating to these language features do not apply to SPARK [SLRM 3.8].

SPARK is also designed to be amenable to static verification of the Liskov Substitution Principle [LSP].

SPARK.3.RIP.4 Implications for standardization

None.

SPARK.3.RIP.5 Bibliography

None.

SPARK.3.LAV Initialization of Variables [LAV]

SPARK prevents this vulnerability.

SPARK.3.LAV.1 Terminology and features

As in Ada.3.LAV.1.

SPARK.3.LAV.2 Description of vulnerability

Ada in Ada.3.LAV.2.

SPARK.3.LAV.3 Avoiding the vulnerability or mitigating its effects

This vulnerability is entirely prevented by use of static information flow analysis [IFA].

SPARK.3.LAV.4 Implications for standardization

None.

SPARK.3.LAV.5 Bibliography

None.

SPARK.3.XYY Wrap-around Error [XYY]

See Ada.3.XYY. In addition, SPARK mitigates this vulnerability through static analysis to show that a signed integer expression can never overflow at run-time [SB 11].

SPARK.3.XZI Sign Extension Error [XZI]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.XZI.

SPARK.3.JCW Operator Precedence/Order of Evaluation [JCW]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.JCW.

SPARK.3.SAM Side-effect and Order of Evaluation [SAM]

SPARK prevents this vulnerability.

SPARK.3.SAM.1 Terminology and features

As in Ada.3.SAM.1.

SPARK.3.SAM.2 Description of vulnerability

As in Ada.3.SAM.2.

SPARK.3.SAM.3 Avoiding the vulnerability or mitigating its effects

SPARK does not permit functions to have side-effects, so all expressions are side-effect free. Static analysis of run-time errors also ensures that expressions evaluate without raising exceptions. Therefore, expressions are neutral to evaluation order and this vulnerability does not occur in SPARK [SLRM 6.1].

SPARK.3.SAM.4 Implications for standardization

None.

SPARK.3.SAM.5 Bibliography

None.

SPARK.3.KOA Likely Incorrect Expression [KOA]

SPARK is identical to Ada with respect to this vulnerability and its mitigation (see Ada.3.KOA) although many cases of "likely incorrect" expressions in Ada are forbidden in SPARK.

SPARK.3.XYQ Dead and Deactivated Code [XYQ]

SPARK mitigates this vulnerability.

SPARK.3.XYQ.1 Terminology and features

As in Ada.3.XYQ.1.

SPARK.3.XYQ.2 Description of vulnerability

As in Ada.3.XYQ.2.

SPARK.3.XYQ.3 Avoiding the vulnerability or mitigating its effects

In addition to the advice of Ada.3.XYQ.3, SPARK is amenable to optional static analysis of dead paths. A dead path cannot be executed in that the combination of conditions for its execution are logically equivalent to *false*. Such cases can be statically detected by theorem proving in SPARK.

SPARK.3.XYQ.4 Implications for standardization

None.

SPARK.3.XYQ.5 Bibliography

None.

SPARK.3.CLL Switch Statements and Static Analysis [CLL]

As in Ada.3.CLL, this vulnerability is prevented by SPARK. The vulnerability relating to an uninitialized variable and the "when others" clause in a case statement is also prevented – see SPARK.3.LAV.

SPARK.3.EOJ Demarcation of Control Flow [EOJ]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.EOJ.

SPARK.3.TEX Loop Control Variables [TEX]

SPARK prevents this vulnerability in the same way as Ada. See Ada.3.TEX.

SPARK.3.XZH Off-by-one Error [XZH]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.XZH. Additionally, any off-by-one error that gives rise to the potential for a buffer-overflow, range violation, or any other construct that could give rise to a predefined exception, will be detected by static analysis in SPARK [SB 11].

SPARK.3.EWD Structured Programming [EWD]

SPARK mitigates this vulnerability.

SPARK.3.EWD.1 Terminology and features

As in Ada.3.EWD.1

SPARK.3.EWD.2 Description of vulnerability

As in Ada.3.EWD.2

SPARK.3.EWD.3 Avoiding the vulnerability or mitigating its effects

Several of the vulnerabilities in this category that affect Ada are entirely eliminated by SPARK. In particular: the use of the goto statement is prohibited in SPARK [SLRM 5.8], loop exit statements only apply to the most closely enclosing loop (so "multi-level loop exits" are not permitted) [SLRM 5.7], and all subprograms have a single entry and a single exit point [SLRM 6]. Finally, functions in SPARK must have exactly one return statement which must the final statement in the function body [SLRM 6].

SPARK.3.EWD.4 Implications for standardization

None.

SPARK.3.EWD.5 Bibliography

None.

SPARK.3.CSJ Passing Parameters and Return Values [CSJ]

SPARK mitigates this vulnerability.

SPARK.3.CSJ.1 Terminology and features

As in Ada.CSJ.1.

SPARK.3.CSJ.2 Description of vulnerability

As in Ada.CSJ.3.

SPARK.3.CSJ.3 Avoiding the vulnerability or mitigating its effects

SPARK goes further than Ada with regard to this vulnerability. Specifically:

- SPARK forbids all aliasing of parameters and names [SLRM 6].
- SPARK is designed to offer consistent semantics regardless of the parameter passing mechanism employed by a particular compiler. Thus this implementation-dependent behaviour of Ada is eliminated from SPARK.

Both of these properties can be checked by static analysis.

SPARK.3.CSJ.4 Implications for standardization

None.

SPARK.3.CSJ.5 Bibliography

None.

SPARK.3.DCM Dangling References to Stack Frames [DCM]

SPARK prevents this vulnerability.

SPARK.3.DCM.1 Terminology and features

As in Ada.3.DCM.1.

SPARK.3.DCM.2 Description of vulnerability

As in Ada.3.DCM.2.

SPARK.3.DCM.3 Avoiding the vulnerability or mitigating its effects

SPARK forbids the use of the 'Address attribute to read the address of an object [SLRM 4.1]. The 'Access attribute and all access types are also forbidden, so this vulnerability cannot occur.

SPARK.3.DCM.4 Implications for standardization

None.

SPARK.3.DCM.5 Bibliography

None.

SPARK.3.OTR Subprogram Signature Mismatch [OTR]

SPARK mitigates this vulnerability.

SPARK.3.OTR.1 Terminology and features

See Ada.3.OTR.1.

SPARK.3.OTR.2 Description of vulnerability

See Ada.3.OTR.2.

SPARK.3.OTR.3 Avoiding the vulnerability or mitigating its effects

Default values for subprogram are not permitted in SPARK [SLRM 6], so this case cannot occur. SPARK does permit calling modules written in other languages so, as in

Ada.3.OTR.3, additional steps are required to verify the number and type-correctness of such parameters.

SPARK also allows a subprogram body to be written in full-blown Ada (not SPARK). In this case, the subprogram body is said to be "hidden", and no static analysis is performed by a SPARK Processor. For such hidden bodies, some alternative means of verification must be employed, and the advice of Annex Ada should be applied.

SPARK.3.OTR.4 Implications for standardization

None.

SPARK.3.OTR.5 Bibliography

None.

SPARK.3.GDL Recursion [GDL]

SPARK does not permit recursion, so this vulnerability is prevented [SLRM 6].

SPARK.3.NZN Returning Error Status [NZN]

SPARK is identical to Ada with respect to this vulnerability and its mitigation. See Ada.3.NZN.

SPARK.3.REU Termination Strategy [REU]

SPARK mitigates this vulnerability.

SPARK.3.REU.1 Terminology and features

As in Ada.3.REU.1.

SPARK.3.REU.2 Description of vulnerability

As in Ada.3.REU.2.

SPARK.3.REU.3 Avoiding the vulnerability or mitigating its effects

SPARK permits a limited subset of Ada's tasking facilities known as the "Ravenscar Profile" [SLRM 9]. There is no nesting of tasks in SPARK, and all tasks are required to have a top-level loop which has no exit statements, so this vulnerability does not apply in SPARK.

SPARK is also amenable to static analysis for the absence of predefined exceptions [SB 11], thus mitigating the case where a task terminates prematurely (and silently) owing to an unhandled predefined exception.

SPARK.3.REU.4 Implications for standardization

None.

SPARK.3.REU.5 Bibliography

None.

SPARK.3.LRM Extra Intrinsics [LRM]

SPARK prevents this vulnerability in the same way as Ada. See Ada.3.LRM.

SPARK.3.AMV Type-breaking Reinterpretation of Data [AMV]

SPARK mitigates this vulnerability.

SPARK.3.AMV.1 Terminology and features

As in Ada.3.AMV.1.

SPARK.3.AMV.2 Description of vulnerability

As in Ada.3.AMV.2.

SPARK.3.AMV.3 Avoiding the vulnerability or mitigating its effects

SPARK permits the instantiation and use of Unchecked_Conversion as in Ada. The result of a call to Unchecked_Conversion is not assumed to be valid, so static verification tools can then insist on re-validation of the result before further analysis can succeed [SB 11].

At the time of writing, SPARK does not permit discriminated records, so vulnerabilities relating to discriminated records and unchecked unions are prevented.

SPARK.3.AMV.4 Implications for standardization

None.

SPARK.3.AMV.5 Bibliography

None.

SPARK.3.XYL Memory Leak [XYL]

SPARK prevents this vulnerability.

SPARK.3.XYL.1 Terminology and features

As in Ada.3.XYL.1.

SPARK.3.XYL.2 Description of vulnerability

As in Ada.3.XYL.2.

SPARK.3.XYL.3 Avoiding the vulnerability or mitigating its effects

SPARK does not permit the use of access types, storage pools, or allocators, so this vulnerability cannot occur [SLRM 3]. In SPARK, all objects have a fixed size in memory, so the language is also amenable to static analysis of worst-case memory usage.

SPARK.3.XYL.4 Implications for standardization

None.

SPARK.3.XYL.5 Bibliography

None.

SPARK.3.TRJ Argument Passing to Library Functions [TRJ]

SPARK mitigates this vulnerability.

SPARK.3.TRJ.1 Terminology and features

See Ada.3.TRJ.1.

SPARK.3.TRJ.2 Description of vulnerability

See Ada.3.TRJ.2.

SPARK.3.TRJ.3 Avoiding the vulnerability or mitigating its effects

SPARK includes all of the mitigations of Ada with respect to this vulnerability, but goes further, allowing preconditions to be checked statically by a theorem-prover. The language in which such preconditions are expressed is also substantially more expressive than Ada's type system.

SPARK.3.TRJ.4 Implications for standardization

None.

SPARK.3.TRJ.5 Bibliography

None.

SPARK.3.NYY Dynamically-linked Code and Self-modifying Code [NYY]

SPARK prevents this vulnerability in the same way as Ada. See Ada.3.NYY.

SPARK.3.NSQ Library Signature [NSQ]

SPARK prevents this vulnerability in the same way as Ada. See Ada.3.NSQ.

SPARK.3.HJW Unanticipated Exceptions from Library Routines [HJW]

SPARK prevents this vulnerability in the same way as Ada. See Ada.3.HJW. SPARK does permit the use of exception handlers, so these may be used to catch unexpected exceptions from library routines.

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 #84: The Distributed Systems Annex 1 – Simple client/server

Thomas Quinot, AdaCore

Date: 19 April 2010

Abstract: This is the first in a series of Gems introducing the facilities defined by the optional annex for distributed systems (Annex E) in the Ada Reference Manual. In this introduction, we show how a simple client/server architecture can be implemented easily with the Distributed Systems Annex (DSA).

Let's get started…

Many aspects of software engineering require, or can benefit from, distributed technology:

- Load balancing
- Fault tolerance
- Interconnection between multiple agents

… among others.

In each of these instances, it is useful to enlist the contribution of multiple computers to achieve a certain goal in a coordinated fashion. In a distributed application design, parts of the processing are thus assigned to distinct hosts which communicate in order to provide a given service. In Ada parlance, the fraction of the complete application that is assigned to each host is called a partition.

A distributed design can be implemented using direct calls to communication services provided by the environment, allowing the exchange of data between partitions. However, this is extremely cumbersome and error-prone. Distribution models have therefore been defined, which are sets of high-level abstractions allowing the programmer to express the interactions between components of a distributed application — possibly located on different partitions — in convenient high-level terms.

Distribution models support various communication patterns. The simplest ones support simple message passing. More elaborate models also provide more structured patterns, such as remote subprogram calls (based on the natural abstraction boundaries represented by subprograms) and distributed (remote) objects, extending remote subprogram calls to the case of method calls in an object-oriented design.

The services afforded by distribution middleware (i.e., the implementation of a distribution model) can be made available to the programmer in different ways. Explicit distribution APIs can be used. Alternatively, distribution may be included in the facilities provided by a programming language. Ada 95 and Ada 2005 include such features as part of the optional Annex E of the Reference Manual.

In this first introductory example, we consider a simple application managing a public bulletin board, which we want to make available for posting from several partitions. The DSA allows a service to be offered in a very simple way: you just write a package declaration:

```
package Bulletin_Board is
  pragma Remote_Call_Interface;
  -- This makes the package a Remote Call Interface (RCI),
  -- so the subprograms below are remotely callable.
  -- This pragma enforces some restrictions on the unit to
  -- ensure that any visible subprogram can actually be
  -- called remotely, and in particular that the types
  -- of the parameters are suitable for transport over a
  --  communication link from one partition to another.
  subtype Length is Natural range 0 .. 100;
  type News_Item (
         Author_Length, Message_Length : Length := 0)
    is record
      Author  : String (1 .. Author_Length);
      Message : String (1 .. Message_Length);
  end record;
  type News_Items is
      array (Positive range <>) of News_Item;
  procedure Post (Item : News_Item);
  function Whats_Up return News_Items;
end Bulletin_Board;
```

A simple client can then be written that will just make calls to these subprograms. The fact that these calls may be executed remotely is completely transparent in the code.

```
with Ada.Text_IO;    use Ada.Text_IO;
with Bulletin_Board; use Bulletin_Board;
procedure Post_Message is
   Author, Message : String (1 .. 140);
   Author_Length, Message_Length : Natural;
begin
   Put ("Author name: ");
   Get_Line (Author, Author_Length);
   Put ("Message    : ");
   Get_Line (Message, Message_Length);

   Post (News_Item'
     (Author_Length  => Author_Length,
      Message_Length => Message_Length,
      Author         => Author (1 .. Author_Length),
      Message        => Message (1 .. Message_Length)));
   -- This subprogram call may be remote, but we write it
   -- exactly in the usual way.
end Post_Message;
```

Similarly, a procedure that displays all messages can be written as follows:

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

```
with Bulletin_Board; use Bulletin_Board;
procedure Display_Messages is
begin
  loop
    Put_Line ("----- all messages -----");
    declare
      Contents : constant News_Items := Whats_Up;
    begin
      for J in Contents'Range loop
        Put_Line (Contents (J).Author & " says:");
        Put_Line (Contents (J).Message);
        New_Line;
      end loop;
      delay 2.0;
    end;
  end loop;
end Display_Messages;
```

This procedure can run on the same partition as the one where Bulletin_Board is located (the language requires that each Remote_Call_Interface unit is assigned to exactly one partition). However, since it only uses a visible subprogram declared in Bulletin_Board (Whats_Up), it could also very well run in another partition.

The assignment of units to partitions need not be apparent in sources. The same set of sources can even be used for different partitioning configurations (or used without partitioning to build a monolithic version of the application, in which case there is no distribution overhead at all).

The process of partitioning a DSA application is implementation defined. In GNAT, this is done using the gnatdist tool, and a po_gnatdist configuration file. The syntax for this file is documented in the PolyORB User's Guide.

Here is an example configuration for the bulletin board application:

```
configuration Dist_App is
  pragma Starter (None);
  -- User starts each partition manually
  ServerP : Partition := (Bulletin_Board);
  -- RCI package Bulletin_Board is on partition ServerP
  ClientP : Partition := ();
  -- Partition ClientP has no RCI packages
  for ClientP'Termination use Local_Termination;
  -- No global termination
  procedure Display_Messages is in ServerP;
  -- Main subprogram of master partition
  procedure Post_Message;
  for ClientP'Main use Post_Message;
  -- Main subprogram of slave partition
end Dist_App;
```

After running po_gnatdist on this configuration file, two executables are produced: serverp and clientp. Serverp will loop, displaying all posted messages, and clientp will allow sending a message to the server. This example thus shows how a simple client/server design can be implemented in Ada without any network programming.

In the next Gem we will discuss remote object designs, which allow flexible dynamic communication across partitions.

Gem #85: The Distributed Systems Annex 2 – Distributed Objects

Thomas Quinot, AdaCore

Date: 03 May 2010

Abstract: This is the second in a series of Gems introducing the facilities defined by the optional annex for Distributed Systems (Annex E) of the Ada Reference Manual. In the first installment, we showed how a simple client/server architecture can be implemented easily with the Distributed Systems Annex (DSA). We now introduce distributed objects, which allow dynamic relationships between components of a distributed application.

Let's get started…

In the previous DSA Gem, we showed how subprograms in a package can be made remotely callable using a pragma Remote_Call_Interface (RCI for short). Each RCI unit is present in only one partition of a distributed application, and any call to a subprogram in such a unit made from another partition is transparently handled by the distribution run-time library.

This is sufficient to implement simple client/server communication, where a single partition is identified as the provider of a service (defined by an RCI package) and accepts requests from other partitions. Different services can be provided by different partitions, and services can be clients of one another. However this scheme is inflexible in that a given service can only ever be provided by a single server. Furthermore, the association between services and partitions is static.

In some contexts, however, more flexible interactions between application components are desired: multiple partitions may want to provide the same service, for performance or fault-tolerance reasons; servers may need to call back their clients; finally, direct (peer-to-peer) interactions between partitions may need to be established in a dynamic fashion, without determining in advance (prior to execution) who will interact with whom.

Such a flexible organization can be implemented using distributed objects. In nondistributed object-oriented programming, an object is an entity with an identity (you can reference, or designate it), internal state, and a set of methods that are common to all objects that belong to the same class, and which represent the ways any object of the class can interact with others. In a distributed world, this paradigm is naturally extended by allowing object references to designate objects that are located on another partition.

In the DSA, distributed objects are created using a specific pragma: Remote_Types. When this pragma is applied to a package, certain type declarations have additional semantics specific to distribution. If you declare a tagged limited private type in such a package, and a corresponding access-to-class-wide type, then that access type is a Remote Access to Class-Wide type (or RACW), and is allowed to designate objects that are located on partitions other than the current one.

These remote object references can be passed around as parameters in remote subprogram calls. For example, they can be sent to an RCI package, or retrieved from it, by passing them as parameters in remote subprogram calls.

Methods of remote objects can be called by just writing a regular dispatching call on any primitive operation. All underlying communication is handled transparently by the distribution run-time library.

So let's now assume that we want to allow users of our bulletin board application to exchange direct messages with one another. Each user will instantiate an object of a concrete type derived from the User type:

```
package Chat_Users is
  pragma Remote_Types;
  -- This package declares a remote object type
  type User is abstract tagged limited private;
  -- Remote objects must be tagged, limited, and private
  type User_Ref is access all User'Class;
  -- This is a remote access-to-class-wide type
  function Name (Who : User) return String;
  procedure Say
   (From : User_Ref;
    To   : User;
    What : String);
  -- The controlling formal 'To' determines the object that
  -- calls are sent to.
  -- The recipient object may be remote. Formal parameter
  -- 'From' is a reference to the originating user, and can be
  -- used to call the user back at a later time.
private
  ...
end Chat_Users;
```

Each message posted to the bulletin board can now include a reference to the message author:

```
with Chat_Users;
package Bulletin_Board is
 ...
 type News_Item (Message_Length : Natural) is
   Author  : Chat_Users.User_Ref;
   Message : String (1 .. Message_Length);
 end News_Item;
 ...
end Bulletin_Board;
```

Now each client can create an instance of a concrete type derived from Chat_Users.User, and pass a 'Access to that object to the bulletin board as it posts messages.

```
with Chat_Users;
package Client is
  -- This is a regular package, no pragma needed
  type Myself_Type is new Chat_Users.User
    with null record;
  function Name (Self : Myself_Type) return String;
  procedure Say
   (From : Chat_Users.User_Ref;
    To   : Myself_Type;
    What : String);
end Client;

with Ada.Text_IO; use Ada.Text_IO;
package body Client is
  function Name (Self : Myself_Type) return String is
  begin
    return "Jean-Pierre";
  end Name;
  procedure Say
   (From : Chat_Users.User_Ref;
    To   : Myself_Type;
    What : String)
  is
    pragma Unreferenced (To);
    -- Parameter 'To' is unused within the body. Its purpose
    -- is just to cause dispatching to the appropriate
    -- object instance.
  begin
    Put_Line ("Got a message from " & From.Name);
    -- Dispatching call to Name to retrieve user name
    -- of the sender 'From'
    Put_Line (What);
    -- Display received message.
  end Say;

  Myself : aliased Myself_Type;
  ...
end Client;
```

Other clients can use the Author component retrieved from the bulletin board to directly contact other clients using the Say method:

```
Say
  (From  => Myself'Access,
   To    => Some_Item.Author,
   Message => "I like it!");
```

Arbitrary partition-to-partition interactions can thus be established using distributed objects. More precisely, these are actually normal objects with the additional property that they can be designated from other partitions using special access types (RACWs). RCI units serve as switchboards to initially propagate references to remote objects across partition boundaries. Once these references are disseminated, partitions can interact directly without the mediation of RCIs.

In the next Gem, we will discuss the implementation of mailbox-based message passing using the Distributed Systems Annex.

Gem #87: The Distributed Systems Annex 3 – Mailboxes

Thomas Quinot, AdaCore

Date: 02 June 2010

Abstract: This is the third in a series of gems introducing the facilities defined by the optional annex for Distributed systems (Annex E) of the Ada Reference Manual. In the previous two installments, we introduced the Distributed Systems Annex (DSA). We showed how a client/server architecture can be implemented, and we introduced distributed objects. The present gem shows how asynchronous message passing can be implemented on top of these facilities.

Let's get started…

In the previous two DSA gems, all communication between partitions occurred as subprogram calls: the received message is handled immediately by the receiving partition (the subprogram body is executed), and the caller resumes execution only after the call returns.

In some applications, a different communication pattern is desired. One partition may want to send a message to another and then forget about it; the receiving partition may not be available to process the message at that time, and may want to keep it queued for later processing.

Sending a message in a "fire-and-forget" fashion can be implemented in the DSA using pragma Asynchronous. This pragma, which applies to subprograms and to remote access types, means that the called subprogram does not return any information (it must be a procedure, and may not have any OUT or IN OUT formal parameters), and that the caller is not interested in any exception that might be raised. When this pragma applies to a remote procedure, execution resumes in the calling task immediately after sending the call (without waiting for any confirmation from the receiver). When the pragma applies to an RACW, this extends to all relevant primitive operations (i.e. procedures with no OUT or IN OUT formals).

The message sending capability is thus simply described by a remote access type declaration:

```
package Mailboxes is
   pragma Remote_Types;
   subtype Message_Type is String;
   -- In this simple example, exchanged messages are
   -- just strings, but this could be changed to any other
   -- type, or made a generic type.
   type Mailbox is limited interface;
   -- This is Ada 2005!
   -- Using an interface as the base type allows the
   -- capability to receive a message to be subsequently
   -- imparted on arbitrary objects (they just need to
   -- implement that interface).
   procedure Send_Message (Recipient : access Mailbox;
                                       Message : Message_Type)
     is abstract;

   type Remote_Mailbox is access all Mailbox'Class;
   -- Remote access to mailbox
   pragma Asynchronous (Remote_Mailbox);
   -- Calls to Send_Message will return to the caller without
   -- waiting for any reply from the callee.
end Mailboxes;
```

A very simple implementation of a mailbox is the "active" mailbox, where a dedicated task handles each incoming message:

```
package Mailboxes.Active is
   task type Active_Mailbox is new Mailbox with
      entry Start (Id : Integer);
      entry Send_Message (Message : Message_Type);
   end Active_Mailbox;
   type Active_Mailbox_Acc is access all Active_Mailbox;
   -- Local access type
end Mailboxes.Active;
```

```
with Ada.Text_IO; use Ada.Text_IO;
package body Mailboxes.Active is
   task body Active_Mailbox is
      My_Id : Integer;
   begin
      accept Start (Id : Integer) do
         My_Id := Id;
      end Start;
      Put_Line ("Active_Mailbox #" & My_Id'Img & " starting");
      loop
         accept Send_Message (Message : Message_Type)
           do
            Put_Line ("... got message: " & Message);
         end Send_Message;
      end loop;
   end Active_Mailbox;
end Mailboxes.Active;
```

Note that this implementation could perfectly well be replaced with any other type implementing the Mailbox interface, for example a protected bounded buffer. Any partition can thus create a mailbox on which it will receive messages from others, just by creating an object of type Mailboxes.Active.Active_Mailbox.

Now, another partition that needs to send it a message will need to obtain an RACW designating that mailbox in order to do so, just like you'd need an address to send a postcard. An RCI package can be used as a central clearinghouse to exchange these initial references: the RCI acts as a directory of partitions that can receive messages.

```
with Mailboxes;
package Hub is
   pragma Remote_Call_Interface;
   procedure Register_Listener (
         Id : Integer;
         Ptr : Mailboxes.Remote_Mailbox);
   -- A partition that has created a mailbox registers it here,
   -- associating it with a unique identifier Id.
   function Get_Listener (Id : Integer)
       return Mailboxes.Remote_Mailbox;
   -- A partition that wants to send a message to the mailbox
   -- identified by Id retrieves the corresponding RACW
   -- (previously registered using the above procedure)
   -- by calling this function.
   -- The implementation of this unit can be as simple as an
   -- array of RACWs:
   --   All_Listeners : array (1 .. Max_Mailboxes) of
   --         Mailboxes.Remote_Mailbox;
end Hub;
```

It should be noted that the RCI is used only to initially disseminate references to partitions. The messages themselves are sent directly across partitions. There is no single point of failure or communication bottleneck.

Complete source code for this application (message sender, message receiver, and central hub) is available in subdirectory examples/dsa/mailboxes of the PolyORB source package, or can also be downloaded directly from the online Gem page.

Gem #90: The Distributed Systems Annex 4 – DSA and C

Thomas Quinot, AdaCore

Date: 14 September 2010

Abstract: This is the fourth in a series of Gems introducing the facilities defined by the optional annex for distributed systems in the Ada Reference Manual (Annex E). In the previous installments, we introduced the Distributed Systems Annex (DSA), and we explained how it allows various interaction paradigms to be implemented. In this Gem, we show how these useful tools can be used from a C program.

Let's get started…

The previous DSA Gems showed how components in a pure Ada application can be spread across several partitions and use static or dynamic remote calls to interact. Wouldn't it be nice if other languages such as C could also benefit from these features?

Of course, you can embed C code in an Ada partition just as you would in any nondistributed application. Your C code can also call back to Ada code (as long as the Ada subprograms have the C convention). Remote (RCI) subprograms can thus be called from C. If the call occurs on the partition to which the RCI is assigned, nothing special happens, this is just a regular call. On other partitions, the compiler-generated calling stubs are used, and this is a transparent remote call, just as it would be if it occurred in Ada code: a remote subprogram has nothing special at the call point; all the magic is done in the generated stubs.

This is all well and good, but you still have to write your complete application in Ada, and in particular have the main subprogram of each partition declared in the GNATDIST configuration file.

What if you would like to incorporate DSA client or server code in an existing C application? This can be achieved by combining the DSA with GNAT's stand-alone libraries, a feature allowing an Ada partition to generate a loadable module rather than a full-fledged executable image. Here's how…

Rebuild PolyORB with -fPIC

The "-fPIC" switch instructs the compiler to generate so-called Position Independent Code, that is, code that can be dynamically loaded as a shared library.

In order to have a DSA partition in a stand-alone library, you need to set CFLAGS="-O2 -g -fPIC" in your environment when calling the PolyORB configure script. (The resulting PolyORB build can also be used for normal applications.)

Build your Ada partitions as usual, also with -fPIC

Let's assume for example that your application has a server partition that is fully written in Ada, and a client partition meant for embedding in a C/C++ application as a shared object. The server partition will be built using:

```
po_gnatdist -fPIC xxxx.cfg server_partition
```

Create a dummy main subprogram for the client side

You need to provide a dummy main subprogram for the client partition. You should make this a null library subprogram that has WITH clauses for any package (including RCIs) that you want to reference from the C side.

Also, it may be convenient to include in this closure an "Exports" package containing suitable subprogram declarations for those routines that you want to call from C, with C-compatible argument types, and using pragma Export to give them friendly C names. (Note that this is not specific to the Distributed Systems Annex: such an interface package is typically created any time you need to call Ada code from C code.)

```
with RCI_1;
...
with RCI_n;

with Exports;
procedure Client is
begin
   null;
end Client;
```

Build the client library

This is the crucial point. To build a partition as a stand-alone library instead of a regular executable, special arguments are passed to GNATDIST:

```
po_gnatdist -fPIC -g xxxx.cfg client_partition \
  -bargs rci_1.ali ... rci_n.ali polyorb-dsa_p-partitions.ali \
        -shared -LClientName \
  -largs -shared
```

In this command line, you need to list the ALI files for all RCI packages referenced in your client partition (rci_1.ali .. rci_n.ali), and also the one for the internal RCI polyorb-dsa_p-partitions.ali.

You can replace the name "ClientName" with an arbitrary prefix of your choosing (it is used for some automatically generated symbols, see below).

This will generate a file client_partition, which you can rename to client_partition.so.

Call client library from C code

Once you have your loadable object generated, you can load it from C code using the standard dlopen(3) function.

Symbols from the library can then be obtained using the dlsym(3) function. You first need to retrieve the symbols ClientNameinit and ClientNamefinal from the library.

ClientNameinit corresponds to the elaboration of all Ada units in the library, and should be called once upon module load. This starts the Ada PCS and connects to the DSA name server to retrieve the initial location of RCI units.

ClientNamefinal corresponds to the finalization, and should be called once, just before unloading the module or terminating the application (ClientName here is the prefix you passed on the GNATDIST command line above).

Finally, you can retrieve and call the symbols for RCI subprograms, or any subprogram exported by your Ada units, and call them as though they were normal C routines.

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

Ada-France
attn: J-P Rosen
115, avenue du Maine
75014 Paris
France
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

Ada-Sweden
attn. Rei Stråhle
Rimbogatan 18
SE-753 24 Uppsala
Sweden
Phone: +46 73 253 7998
Email: rei@ada-sweden.org
URL: www.ada-sweden.org

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