ADA USER JOURNAL

Volume 33
Number 4
December 2012

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. The language of the journal is English.

Although the title of the Journal refers to the Ada language, related topics, such as reliable software technologies, are welcome. More information on the scope of the Journal is available on its website at *www.ada-europe.org/auj*.

The Journal publishes the following types of material:

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

Further details on our approach to these are given below. More complete information is available in the website at *www.ada-europe.org/auj*.

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

Proceedings and Special Issues

The *Ada User Journal* is open to consider the publication of proceedings of workshops or panels related to the Journal's aims and scope, as well as Special Issues on relevant topics.

Interested proponents are invited to contact the Editor-in-Chief.

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. Our readers need not surf the web or news groups to find out what is going on in the Ada world and in the neighbouring and/or competing communities. We will 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 electronically. Authors are invited to contact the Editor-in-Chief by electronic mail to determine the best format for submission. The language of the journal is English.

Our refereeing process aims to be rapid. Currently, accepted papers submitted electronically are typically published 3-6 months after submission. Items of topical interest will normally appear in the next edition. There is no limitation on the length of papers, though a paper longer than 10,000 words would be regarded as exceptional.

Editorial

The reader may remember that last June the Ada User Journal published a press release on the completion of the design of Ada 2012 and the submission to ISO for approval. I am glad to inform that this formal process concluded this December, with the official publication of the new version of the Standard. The Standard received the approval vote of 18 members of ISO, without comments or rejection ballots, in a swift process that between submission and approval only took around six months to proceed. This was only possible due to the high-quality of the work carried by the ARG and WG9 prior to submission, and the effort to quickly act in the ISO process.

I am sure that you will join me in congratulating both groups for this effort, and for the successful outcome.

As for the contents of the issue you are reading, we provide a new chapter of the Rationale for Ada 2012, which deals with improvements on several areas of the language, such as iterators, storage pools, new restrictions, a new profile (No_Implementation_Extensions), etc.

Afterwards, we publish two papers derived from the industrial track of Ada-Europe 2012. In the first paper, a group of authors from Embraer, Brazil, and Rapita Systems, UK, present the experience of obtaining worst-case execution times and worst case stack usage for a DO-178B level A Flight Control System. In the second paper, authors from GMV, Spain, and Universidad Politécnica de Madrid, Spain, present a study on the integration of Hardware/Software co-design in the ASSERT model-driven engineering process.

The issue continues with the return of the section on Ada User Guides, presenting a paper on how to program 8-bit AVR Micro-Controllers with Ada, a contribution from Pablo Rego, from Embraer, Brazil. After a successful session at Ada-Europe 2012 on using Ada for Lego Mindstorms and Arduino-based devices, this is another contribution to promote the use of Ada for programming small embedded systems.

And as usual the News Digest, Calendar and Forthcoming Events sections complete the issue. In the latter, the reader will find information on several events related to Ada which will take place in 2013: the Ada Developer Room at the Free and Open source Software Developers' European Meeting taking place February 3 in Brussels, Belgium; the 16th International Real-Time Ada Workshop, which will take place April 17-19, in York, UK; the 18th International Conference on Reliable Software Technologies – Ada-Europe 2013 that will take place June 10-14, in Berlin, Germany; and the SIGAda's High Integrity Language Technology conference, taking place fall 2013, in Pittsburgh, USA. Undoubtedly, a full year.

A final note to the "try and beat me" poster that the issue displays; after the announcement of the winners of the AdaWay student contest, and the posting online of the sources of this implementation, an open ended challenge is now open: any student team can attempt to improve over the reference implementation, on any of the evaluation criteria. Good Luck!.

Luís Miguel Pinho
Porto
December 2012
Email: AUJ_Editor@Ada-Europe.org

Quarterly News Digest

Jacob Sparre Andersen

Jacob Sparre Andersen Research & Innovation. Email: jacob@jacob-sparre.dk

Contents

Ada Semantic Interface Specification (ASIS)

Is ASIS being utilized as much as it can be?

From: Joseph Wisniewski <wisniewski.ru@gmail.com>
Date: Tue, 25 Sep 2012 18:26:21 -0700
Subject: Is ASIS being utilized as much as it can be?
Newsgroups: comp.lang.ada

What are general views on how well ASIS is being used?

If you are a vendor, do you see that your clients are adequately aware of what ASIS can do for them? Do you not have 'critical mass of interest' to implement features/annexes post Ada 95? I know that that is an issue with some subset of the vendors.

As users, do you see ASIS as "too difficult" to learn and implement? Is there an impression (or reflection of reality) that there just a general lack of developers that are able to "do ASIS" and develop ASIS tools?

Were there programs perhaps that developed ASIS (V1.0 - Ada 83) applications and found that the "port" to V2.0 and greater was "too much work or too difficult"?

Ada and Education

AdaTutor on the web

From: Zhu Qun-Ying <zhu.qunying@gmail.com>
Date: Fri, 28 Sep 2012 22:50:00 -0700
Subject: ANN: AdaTutor on the Web - done
Newsgroups: comp.lang.ada

I have finally finished the conversion of the AdaTutor in my blog space:

http://zhu-qy.blogspot.com/2012/08/adatutor.html

[See also "AdaTutor is back online", AUJ 32-3 (September 2012), p. 136. —sparre]

From: Christoph Grein <christ-usch.grein@t-online.de>
Date: Sat, 29 Sep 2012 03:22:41 -0700
Subject: Re: ANN: AdaTutor on the Web - done
Newsgroups: comp.lang.ada

This text in the tutorial about UC is nonsense:

“One use of Ada.Unchecked_Conversion might be to allow us to and two Integers. (Ada 95 allows us to and two objects of a modular type, but not two Integers.) Some Ada compilers come with a package that enables us to and two Integers, but many compilers have no such package. Suppose that types Integer and Boolean occupy the same amount of storage. If our program says with Ada.Unchecked_Conversion; we could write

```
function Int_To_Bool is new
  Ada.Unchecked_Conversion(Integer,
  Boolean);
function Bool_To_Int is new
  Ada.Unchecked_Conversion(Boolean,
  Integer);
function "and" (Left, Right : in Integer)
return Integer is
begin
  return Bool_To_Int(Int_To_Bool(Left) and
         Int_To_Bool(Right));
end "and"; ”
```

Integer and Boolean have never the same size. Boolean'Size = 1. "and" will pick either the MSB or the LSB of Left and Right, depending on compiler resp. hardware.

(Ada 83 was a bit unclear about the meaning of 'Size, there are differences in the behaviour of compilers. Ada 95 fixed that - in a way that some find awkward.)

UC between Integer and Natural might work or include a multiplication/division by 2 because Natural'Size=Integer'Size-1. I have been bitten by UC between objects of types with different sizes when porting some legacy Ada 83 code to a different hardware and compiler. Be careful: Stand-along objects of subtype Natural and of Integer have of course the same size - what matters for UC is the subtype's size, which is given by 'Size - this is why GNAT has an attribute 'Object_Size.

From: Shark8 <onewingedshark@gmail.com>
Date: Sun, 30 Sep 2012 21:39:46 -0700
Subject: Re: ANN: AdaTutor on the Web - done
Newsgroups: comp.lang.ada

One little nit to pick, in the recursion example (http://zhu-qy.blogspot.com/2012/08/adatutor-recursion.html) factorial is a bad choice to illustrate recursion as it is worse, both in speed and size [due to its linear call-nature], than the iterative/for-loop -- a better candidate would be the power function as it presents a logarithmic call-nature… though I fully expect "**" to be optimized even beyond the simple recursive-function.

Ada-related Resources

Introduction to Glade 3

From: Françs Fabien <francois_fabien@hotmail.com>
Date: Mon, 17 Sep 2012 20:31:21 +0200
Subject: Hello world with glade 3
URL: http://wiki.ada-dk.org/building_gui_with_glade_3

I read on this list some posts for starting with glade 3. ("hello world" with glade 3). To gain more visibility on the web, I put a summary on Ada-DK wiki:

http://wiki.ada-dk.org/building_gui_with_glade_3

Remote-controlled robot using XBees and Ada

From: Tero Koskinen
Date: October 24, 2012
Subject: Remote-controlled robot using XBees and Ada
URL: http://arduino.ada-language.com/remote-controlled-robot-using-xbees-and-ada.html

As a continuation from my motor shield article [1], I put my motor shield and Arduino combination on wheels and added a wireless shield to go with them.

[1] http://arduino.ada-language.com/running-a-motor-with-motorshield-and-pulse-width-modulation-pwm.html.

[Tero takes us through the whole process of selecting the parts, putting the robot together and writing the source text to control it. —sparre].

IRC channel

From: Genro Kane <wmrg100@niestu.com>
Date: Wed, 7 Nov 2012 17:44:39 -0800
Subject: [ANN] Ada IRC channel on Freenode
Newsgroups: comp.lang.ada

About twice a year we try to advertise the #Ada channel on the Freenode IRC network. Celebrating its eleventh birthday next month, the channel continues to be active and friendly. These days it averages about 60 users at a time, large enough to support lively and informative discussions but small enough so it's not a madhouse.

Topics range all over the map, from building the latest GNAT to writing an OS in Ada to daily Ada programming issues to how to use PolyOrb to use the Distributed Systems Annex. The stated topic is discussing Ada in the context of free and open-source software, but commercial users are equally welcome.

So fire up your favorite IRC client and come join us! The network is homed at irc.freenode.net, but has servers all over the world. Visit www.freenode.net on the web for details. Hope to see you soon!

Controlling 4-digit 7-segment display with AVR-Ada

From: Tero Koskinen
Date: November 9, 2012
Subject: Controlling 4-digit 7-segment display
URL: http://arduino.ada-language.com/controlling-4-digit-7-segment-display.html

A seven-segment display is nice for showing a single number. And 4-digit display is 4 times as nice since it can show four numbers.

However, using 4-digit version is also somewhat more complicated. The 4-digit 7-segment display (which I use) is done so that each digit will have same set of leds turned on at the same time when the digits are turned on. This means that if we want to show a different number in a different digit, we need to turn other digits off. Fortunately for us, other leds won't turn off immediately, so if you do turning on and off quick enough, you can set the display so that each digit shows a different number.

[Tero takes us through the whole process of connecting the display to the microcontroller and writing the source text to control it. —sparre]

Directories of Open Source software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: November 30 2012
Subject: Directories of Open Source software
IRC: #Ada on irc.freenode.net

Free(code): 76 projects [1]

Ohloh: 73_268 source files [2]

[1] http://freecode.com/search?page=1&with=2880

[2] http://code.ohloh.net/search?s=is&fl=Ada&filterChecked=true

Repositories of Open Source software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: November 30 2012
Subject: Repositories of Open Source software
IRC: #Ada on irc.freenode.net

ada.cx: 8 repositories [1]

AdaForge: 7 repositories [2]

Bitbucket: 29+ repositories [3,4]

Codelabs: 9 repositories [5]

GitHub: 289 repositories [6]
60 developers [7]

Rosetta Code: 549 examples [8]
24 developers [9]

Sourceforge: 200+ repositories [10]

[1] http://git.scm.ada.cx/

[2] http://forge.ada-ru.org/adaforge

[3] https://bitbucket.org/repo/all/relevance?name=binding&language=ada

[4] https://bitbucket.org/repo/all/relevance?name=ada&language=ada

[5] http://git.codelabs.ch/

[6] https://github.com/search?q=language%3AAda&type=Repositories

[7] https://github.com/search?q=language%3AAda&type=Users

[8] http://rosettacode.org/wiki/Category:Ada

[9] http://rosettacode.org/wiki/Category:Ada_User

[10] http://sourceforge.net/directory/language%3Aada/

Ada group on Google+

From: Thomas Løcke <tl@ada-dk.org>
Date: December 8, 2012
Subject: New Ada community at Google+
URL: http://identi.ca/notice/98337737

New !Ada community at Google+ http://ur1.ca/bwub4

Ada-related Tools

Generic Image Decoder

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Sat, 8 Sep 2012 11:22:36 -0700
Subject: Ann: Generic Image Decoder v.02
Newsgroups: comp.lang.ada

I have the pleasure to announce the second release of GID - the Generic Image Decoder.

URL: http://gen-img-dec.sf.net/

New in V.02

GID is now able to inform about image orientation, e.g. JPEG EXIF tag which is usual on digital cameras.

See the To_BMP demo for an example of how this information is used to rotate target image.

About GID

The Generic Image Decoder (GID) is an Ada package for decoding a broad variety of image formats, from any data stream, to any kind of medium, be it an in-memory bitmap, a GUI object, some other stream, arrays of floating-point initial data for scientific calculations, a browser element, a device,…

Animations are supported.

Features

- Standalone (no dependency on other libraries, bindings,…)
- Unconditionally portable code: OS-, CPU-, compiler- independent code.
- Multi-platform, but native code built
- Task safe
- Endian-neutral
- Use of generics and inlining at multiple nesting levels for fast execution
- Free, open-source

Currently supported formats are: BMP, GIF, JPEG, PNG, TGA.

[See also "Generic Image Decoder v.01", AUJ 31-3 (September 2011), p. 157. —sparre]

YAMI4

From: Maciej Sobczak <maciej@msobczak.com>
Date: Mon, 10 Sep 2012
Subject: YAMI4 1.6.0 released with message queue server (implemented in Ada) and Objective-C library bindings
URL: http://inspirel.com/yami4/

YAMI4 is a set of messaging libraries designed for distributed systems with particular focus on control and monitoring systems. The major features of YAMI4 are:

* peer-to-peer messaging

* support for message priorities

* built-in support for load balancing and automatic fail-over

* support for publish-subscribe messaging

* support for isolated memory partitions (intended for embedded and critical systems)

* support for real-time development with comprehensive range of timeout features

* high performance and scalability with non-blocking I/O

* small memory and resource footprint

* no dependencies on other libraries

[See also "Inspirel — YAMI4 1.4.0", AUJ 32-4 (December 2011), p. 141. —sparre]

Dequesterity

From: Brad Moore <brad.moore@shaw.ca>
Date: Sat, 15 Sep 2012 10:43:59 -0600
Subject: ANN: Dequesterity 1.3
Newsgroups: comp.lang.ada

I am pleased to announce the availability of Dequesterity, version 1.3.

Dequesterity is a set of Ada 2005 generics that provide various forms of general purpose buffer containers. Buffers may be used as deques, queues, ring buffers, stacks, double ended stacks, vectors, priority queues, and similar abstractions.

There are various concurrent buffers, priority buffers, streaming buffers, remote buffers. In fact there are now over 100 buffer packages to choose from.

Some of the new features of this release include;

- Added priority buffers. This is an intermediate buffer class that can be combined with lower simple buffer classes, and optionally with higher level concurrency buffers to generate a buffer with the desired features.

- Converted a number of class-wide subprograms to primitive subprograms. This makes these calls easier to use, including object prefix notation.

- Fixed bug in Replace_Element that caused crash in indefinite bounded buffers.

- Invalidate cursors when appropriate in all buffer types

- Added Swap primitive to swap two elements in a buffer.

NOTE: The Definite Priority Buffer generics do not currently compile under GNAT, due to a compiler bug. However, they do compile with the ICC Ada 2005 compiler.

The Indefinite Priority buffer generics, however do compile under GNAT, and since any definite type can be used to instantiate an indefinite priority buffer, these can be used as a work around until a version of GNAT is available that addresses the compiler bug.

This release and older releases may be downloaded from;

https://sourceforge.net/projects/dequesterity/files/

[See also "Dequesterity", AUJ 33-2 (June 2012), p. 79. —sparre]

Tcl/Tk

From: Patrick <patrick@spellingbeewinnars.org>
Date: Tue, 18 Sep 2012 04:56:08 -0700
Subject: Tcl/TK + Ada
Newsgroups: comp.lang.ada

This is probably just a continuation of the recent Ada GTK vs QTbinding thread but I didn't want to take a chance of thread-jacking.

So I am working on a daemon that will be written in Ada and controlled by a slew of small simple commands. I am planning on controlling it through a Tcl/Tk GUI. I am wondering if this is a good design decision? Has anyone done something like this?

Tcl is not going to be as reliable as Ada but if the code base is small (and Tcl was designed for small code bases) then I should be okay right?

There is an Ada binding:

http://tcladashell.sourceforge.net/index.htm

Am I walking into any pitfalls here?

From: Thomas Løcke <tl@ada-dk.org>
Date: Tue, 18 Sep 2012 14:42:25 +0200
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

If the daemon is controlled by a "slew of simple commands", why not just control it from the command line interface? That way your daemon is also controllable from scripts, making it easier for sysadmins to manage it.

Personally I can't stand daemons that are managed from a GUI, as a GUI usually limits my ability to control the daemon however I want.

Server software without a decent CLI is a dreadful thing to work with.

From: Patrick <patrick@spellingbeewinnars.org>
Date: Tue, 18 Sep 2012 05:51:39 -0700
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

Yep I had similar experiences with the cherokee web server. I didn't want to clutter the list but as per usual I can't be short and clear.

Basically the daemon is an scientific instrument control server. I need a GUI for data visualization. The server would maintain sanity and abstract the PC's hardware. So for instance a command could end up being sent over RS232, ethernet or GPIB and if a command was sent out of order, like some sort of initialization after the instrument was already running, then the server could ignore it.

From: Simon Wright <simon@pushface.org>
Date: Tue, 18 Sep 2012 18:36:54 +0100
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

When you say "daemon" I imagine a process which runs continually, and needs to be controlled using separate, probably short-lived, processes run from e.g. the command line and which interact with the daemon process using some sort of inter-process comms.

The natural way of writing a TASH/Tk app such as you describe has the GUI as part of the app, and that's not how daemons work.

Given that, even though I'm one of TclAdaShell (aka TASH)'s maintainers, I don't see any advantage to writing the control process using TclAdaShell (aka TASH) vs the web approach, where connecting via a browser corresponds to the short-lived "process".

I do have a minimal web server EWS[1], but it doesn't hold your hand at all with connecting web pages via e.g. AJAX to an Ada application. You can do it, but it's a lot of work.

In distinction, AWS from AdaCore has a lot of support for this sort of thing, and is definitely worth a look.

[1] https://sourceforge.net/projects/embed-web-srvr/

From: Patrick <patrick@spellingbeewinnars.org>
Date: Tue, 18 Sep 2012 11:57:21 -0700
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

I am still really in the planning stage right now…but when I say daemon I mean a process that stays alive and accepts instructions. I was planning on using a port to receive on.

So let's say we were controlling an optical spectrometer, we might have commands like:

set_wavelength

set_high_voltage

set_stirrer

plot

etc.

So take the plot command for instance. I want the command to send an instruction and arguments to the daemon and then return right away. The plotting could take hours or days so the daemon could be responsible for it after that point.

If I had all these simple little commands to control the daemon I thought I could also build a gui for it to visualize the data being collected.

From: Simon Wright <simon@pushface.org>
Date: Tue, 18 Sep 2012 20:30:36 +0100
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

> […]

OK. You could build the GUI in straight Tcl/Tk and have it talk to the daemon's port (on Linux, I guess that'd be a UNIX-domain socket).

> […]

Yes, indeed; but it'd need to be on the user side, not the daemon's, so you'd need a protocol to request the current stats/return them (2-way comms).

Equally you could use the same architecture but with Gtk/Ada.

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Tue, 18 Sep 2012 19:23:19 -0400
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

Mentally, I would consider "plot" to be an analysis operation performed on collected data. As such, it would NOT be a responsibility of the daemon performing data collection.

The various "set_" operations, OTOH, implement configuration changes regarding what data is collected, and thereby seem viable as command-line controls (creating a GUI later to invoke those controls is another option, with the GUI independent of the daemon itself).

From: john@peppermind.com
Date: Wed, 19 Sep 2012 03:02:05 -0700
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

Like others said, you should provide all means to control the behavior of the daemon via command line.

Apart from that, Tcl/Tk should be fine, but you might also want to consider building a browser-based interface, e.g. using AWS.

It's a bit more complicated, but it's "modern" (whether you like it or not) and does have advantages. For example, by using SSL and basic authentication a browser-based interface could easily be used for remote-controlling the daemon if security requirements are not very high.

From: Pascal Obry <pascal@obry.net>
Date: Wed, 19 Sep 2012 13:17:45 +0200
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

> […]

Or use a client certificate based authentication which is the highest security level when dealing with Web applications. Support for this has recently been added into AWS. Worth looking at the documentation I would say.

From: john@peppermind.com
Date: Wed, 19 Sep 2012 06:05:16 -0700
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

> […]

Is your daemon not allowed to execute external processes? If not, there are all kinds of very powerful command-line utilities you could use, e.g. gnuplot.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 19 Sep 2012 17:12:39 +0200
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

[…]

Don't want to spoil the party, but GIMP is based on GTK. So is AICWL [1].

It has oscilloscope widget which can be used for plotting, both off-line and in-the-loop. Since it is a plain GTK widget, you can put into whatever window you wished.

[1] http://www.dmitry-kazakov.de/ada/aicwl.htm#12.5

From: Leonid Dulman <leonid.dulman@gmail.com>
Date: Fri, 21 Sep 2012 01:54:26 -0700
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

If you want to use Ada with Tcl/Tk, try VAD (Visual Ada Developer) from http://users1.jabry.com/adastudio/index.html

From: Jerry <lanceboyle@qwest.net>
Date: Fri, 21 Sep 2012 16:23:18 -0700
Subject: Re: Tcl/TK + Ada
Newsgroups: comp.lang.ada

And there's PLplot, a linkable library with fine ;-) Ada bindings and extensions.

http://plplot.sourceforge.net/

Ada 2012 OpenGL binding

From: Shark8 <onewingedshark@gmail.com>
Date: Thu, 18 Oct 2012 10:16:51 -0700
Subject: Announcement: OpenGL Binding w/ Ada-2012.
Newsgroups: comp.lang.ada

I've recently completed the initial binding for OpenGL to Ada 2012. It's available at http://github.com/OneWingedShark/TAO-GL

Some notes:

1) Currently all the functions from OpenGL 1.1 are included; this is because I didn't realize at the time how dated the C-headers I started out with were - though it makes little difference in the overall amount of work which is mostly reading documentation and typing/subtyping enumerations for the parameters.

2) The preconditions are at present mostly minimalistic (just the easy/straightforward preconditions), though all of them should have the correct restrictions regarding the "raises error between glBegin/glEnd" restriction.

3) There are a few points where the binding is not nearly thick enough for my taste, mostly because I'm unsure of how to correctly translate the function [parameters esp] into nice Ada; this usually results in an exposed System.Address parameter.

GNAT licensing questions

From: Aditya Siram <aditya.siram@gmail.com>
Date: Sat, 20 Oct 2012 10:18:18 -0700
Subject: Licensing Questions
Newsgroups: comp.lang.ada

Having looked at the Alioth benchmarks I'm looking into Ada as an alternative to C++ but I'm totally confused by the licensing.

It seems like there's two entities GNU and AdaCore both offering a compiler/runtime but with different restrictions so please excuse these simple questions:

- How are the two products different? Are they just the same codebase with different support models?
- Can I develop commercial closed-source libraries and software using the GNU version of GNAT?
- Will there be compatibility issues between GNU GNAT and AdaCore GNAT beyond missing Ada 2012 features?
- Is AdaCore the only company working on GNAT right now?
- Are there performance differences between the two compilers?

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Sat, 20 Oct 2012 20:03:36 +0200
Subject: Re: Licensing Questions
Newsgroups: comp.lang.ada

Same code base, but different support models, indeed.

- FSF GNAT comes with no support, and a delay of about one year lagging behind GNAT Pro/GNAT GPL ;
- GNAT GPL comes with no support, but benefits from more reactive updates than FSF GNAT, while still lagging behind GNAT Pro;
- GNAT Pro comes with full support and a very reactive team of long time experienced professional Ada developers.

> - Can I develop commercial closed-source libraries and software using the GNU version of GNAT?

- With FSF GNAT which comes with the runtime library exception: yes (*)
- With GNAT GPL: no
- With GNAT Pro: yes (**)

(*) Beware that some some other conditions may make your applications covered by the GPL.

(**) It will even give you the right to use the otherwise GPL additional libraries, for any purpose you want, either commercial, open, closed source.

> - Will there be compatibility issues between GNU GNAT and AdaCore GNAT beyond missing Ada 2012 features?

FSF GNAT is lagging, but that's a matter of delay and patience. GNAT GPL is more up to date. GNAT Pro is a lot more up to date than the two former. There are compatibility issues with AUnit (AUnit is to Ada, what JUnit is to Java), and both are incompatible.

> - Is AdaCore the only company working on GNAT right now?

As far as I know, yes, except there are people working on GNAT FSF too.

> - Are there performance differences?

No. Except if you need support for no-runtime variant or any other kind of specific runtime meeting specific requirement, then you will need GNAT Pro. Otherwise, both provides comparable performances.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Sat, 20 Oct 2012 21:25:50 +0300
Subject: Re: Licensing Questions
Newsgroups: comp.lang.ada

I'll take a shot at answering you. I'm not a lawyer, and my understanding may be incorrect or incomplete, but here is how I understand it.

> It seems like there's two entities GNU and AdaCore both offering a compiler/runtime but with different restrictions

The GNAT compiler is available with three different licensing schemes:

- GNAT Pro from AdaCore. Expensive, but you can distribute your Ada programs in binary form (for free or for a price) without distributing source. And you get support from AdaCore.
- GNAT GPL from AdaCore. Free beer, but with pure GPL, so can be used only for developing GPL programs.
- GNAT from the Free Software Foundation (FSF; I think this is what you call "GNU GNAT"). Free beer, and the compiler itself is under GPL, but the programs that you develop with the compiler are not, because the libraries use a modified version of GPL that lets you use the libraries in non-GPL programs.

I use FSF GNAT to compile SW for which I sell binary-only licences in the traditional fashion.

> - How are the two products different? Are they just the same codebase with different support models?

The codebase is roughly the same; GNAT Pro is most advanced and is frequently updated. GNAT GPL is a snapshot of GNAT Pro taken yearly (more or less). FSF GNAT is maintained by the community but receives updates from AdaCore (tracking GNAT Pro) now and then.

> - Can I develop commercial closed-source libraries and software using the GNU version of GNAT?

With the FSF GNAT, yes. The Debian platform has excellent support for FSF GNAT, thanks mainly to Ludovic Brenta. For MS Windows, MinGW provides FSF GNAT (as I understand it). It seems to work, too.

> - Will there be compatibility issues between GNU GNAT and AdaCore GNAT beyond missing Ada 2012 features?

On the Ada source-code level, for application programs, I don't expect any compatibility problems.

On the internal level, the ASIS tool-set depends on GNAT internals and must be closely matched to GNAT, version by version. You cannot use the GNAT GPL version of ASIS to access compiled code from FSF GNAT. Also, I would hesitate to mix object code from different GNAT versions in the same link.

> - Is AdaCore the only company working on GNAT right now?

Yes, for the PC targets, as far as I know. There have been some companies that have developed off-shoots of GNAT for embedded systems, for example the XGC company in the UK.

> - Are there performance differences between the two compilers?

Could be, since GNAT Pro is the "wavefront" and so probably has more evolved optimizations than the other versions.

From: Simon Wright <simon@pushface.org>
Date: Sat, 20 Oct 2012 20:20:16 +0100
Subject: Re: Licensing Questions
Newsgroups: comp.lang.ada

Most of the updates to FSF Ada are from AdaCore contributing a massive input at the start of each release (4.6, 4.7, 4.8); there are contributions from others in between whiles, but still AdaCore contribute a very great deal.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sat, 20 Oct 2012 23:28:21 +0200
Subject: Re: Licensing Questions
Newsgroups: comp.lang.ada

I'm not going to repeat the good answers from Yannick and Niklas (BTW, thanks to both for mentioning me :)) but in case you didn't know, there are other Ada compilers besides GNAT. The full and complete answer is in the Ada Programming wikibook:

http://en.wikibooks.org/wiki/Ada_Programming/Installing

The rest of the book is a good read, too, and will answer many more questions.

Simple components for Ada

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 20 Oct 2012 21:09:24 +0200
Subject: ANN: Simple components for Ada v3.21
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.

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

[See also "Simple Components for Ada", AUJ 33-2 (June 2012), p. 78. —sparre]

Qt5Ada

From: Leonid Dulman <leonid.dulman@gmail.com>
Date: Tue, 30 Oct 2012 10:08:25 -0700
Subject: Announce : Qt5Ada version 5.0.0 beta1 free edition
Newsgroups: comp.lang.ada

Qt5Ada is Ada-2012 port to Qt5 framework.

It supports GUI, SQL, Multimedia, Web, Network and many others things.

Qt5Ada for Windows and Linux (Unix) is available from

http://users1.jabry.com/adastudio/index.html

Zip-Ada

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Sun, 4 Nov 2012 18:37:49 -0800
Subject: Ann: Zip-Ada v.44
Newsgroups: comp.lang.ada

I am pleased to announce a new version of the Zip-Ada library [1].

Major changes in the last version:

- The library provides now a complete toolset for managing Zip archives (creating, updating, extracting)
- Zip_Streams: All methods now with pointer-free profiles
- Zip.Create:
  - new Add_Compressed_Stream for copying entries from an archive to another one
  - file modification date and read-only attribute can be passed to Add_File
  - archive creation date is used when entries are added which are from a memory buffer (Add_String)
  - compression method can be changed "on the fly", before adding new entries

Some features:

- full sources are in Ada (no binding)
- decompression for all Zip sub-formats up to BZip2
- compression for all Zip sub-formats up to Deflate
- unconditionally portable
- input and output can be any stream (file, buffer,…) for archive creation as well as data extraction.
- task safe
- endian-neutral

The zipada44.zip archive contains:

- The full library sources (inside one directory: Zip_Lib), in Ada 95, using only Ada 95 standard's libraries
- Some command-line demo / tools:
- ZipAda, a zipping tool
- UnZipAda, an unzipping utility
- Comp_Zip, compares two Zip files
- Find_Zip, searches a text string through contents of a Zip file
- ReZip.adb, optimizes compression of Zip archives

NB: the ZipAda, UnZipAda, ReZip use a few non-Ada 95 library items.

[1] http://unzip-ada.sf.net/

[See also "Zip-Ada v.41 (beta)", AUJ 32-3 (September 2011), p. 137. —sparre]

Ada BFD

From: Stephane Carrez <Stephane.Carrez@gmail.com>
Date: Sun, 11 Nov 2012 10:45:36 +0100
Subject: [ANN] Ada BFD 1.0 is available
Newsgroups: comp.lang.ada

Ada BFD is an Ada binding for the GNU Binutils BFD library. It allows to read binary ELF, COFF files by using the GNU BFD. The Ada BFD library allows to:

- list and scan the ELF sections of an executable or object file,
- get the content of the ELF sections,
- get access to the symbol table,
- use the BFD disassembler

Version 1.0 of this Ada binding library is now available at [1].

This new release bring the following changes:

- Fixed installation of library
- Added examples for BfdAda
- Add support to use the disassembler

For an introduction on using Ada BFD, you may read the article: Reading a program symbol table with Ada BFD [2]

[1] http://code.google.com/p/ada-bfd/

[2] http://blog.vacs.fr/index.php?post/2012/11/03/Reading-a-program-symbol-table-with-Ada-Bfd

Anet

From: Reto Buerki <reet@codelabs.ch>
Date: Thu, 15 Nov 2012 11:20:26 +0000
Subject: ANN: Anet version 0.2
Newsgroups: comp.lang.ada

I'm proud to announce the first official release of Anet, a networking library for the Ada programming language.

The project website is at [1], the current release 0.2 can be downloaded from [2]. An example on how to use Anet to implement client/server-applications is provided on the project page as well.

Anet currently provides the following features:

- BSD socket implementation
- High abstraction level
- Extendable socket type hierarchy
- Socket receiver tasks (Stream and Datagram)
- Ada type serialisation/deserialisation over sockets
- Supported socket families
  - IPv4 (AF_INET)
  - IPv6 (AF_INET6)
  - Packet (AF_PACKET)
  - UNIX domain (AF_UNIX)
  - Netlink (AF_NETLINK)
- Supported socket modes
  - Stream (TCP)
  - Datagram (UDP)
- Support for IPv4/IPv6 multicast
- UDP/IPv4 packet creation and validation
- Binding to the Linux Packet Filter (LPF) system

[1] http://www.codelabs.ch/anet/

[2] http://www.codelabs.ch/download/

OpenID authentication for AWS

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Sat, 01 Dec 2012 14:15:26 +0100
Subject: ANN: OpenID authentication for AWS
Newsgroups: comp.lang.ada

I'm pleased to announce that my OpenID client for AWS (forked from the ASF OpenID implementation) is in a usable state:

http://repositories.jacob-sparre.dk/openid-client

The repository includes a demonstration application (which can be seen running at <https://jaws.adaheads.com/>).

Help writing documentation is welcome. The same goes for help making the remaining ASF-originating files more "Ada-like".

[See also "Ada Server Faces", AUJ 33-2 (June 2012), p. 76. —sparre]

Ada-related Products

VectorCAST validated against ACATS 3.0

From: Vector Software
Date: 2012-08-13
Subject: VectorCAST validated against the Ada Conformity Assessment Test Suite (ACATS) 3.0
URL: http://www.vectorcast.com/news/2012/pr_vector-supports-latest-ada.php

Vector Software remains committed to the Ada software development community.

Providence, RI – 08/13/2012 - Vector Software, the leading provider of software solutions for testing safety critical applications, announced today that its VectorCAST embedded software testing platform has been validated against version 3.0 of the Ada Conformity Assessment Test Suite (ACATS).

ACATS provides the official tests used to check conformity of an Ada implementation with the Ada programming language standard. ACATS 3.0 is the test suite for the current version of Ada as defined by the joint ISO/ANSI standard ISO/IEC 8652:1995/Amd 1:2007.

Vector Software's VectorCAST/Ada is an integrated software test solution that significantly reduces the time, effort, and cost associated with testing Ada software components necessary for validating critical embedded systems.

VectorCAST/Ada parses source code and invokes code generators to automatically create the test code (stubs and drivers) required to construct a complete, executable test harness. Once the test harness is constructed, utilities can be used to build and execute test cases and report static measurements. Test data is maintained separately from the test harness enabling easy automatic regression testing.

Additionally, Vector Software's code coverage analysis tool VectorCAST/Cover, has been updated to support Ada 2005. VectorCAST/Cover analyzes the completeness of system tests ensuring that applications are not released with untested code. With VectorCAST/Cover developers can gauge the effectiveness of their test efforts by identifying which areas of an application were exercised during a test run.

"Ada has long been recognized as a highly effective language for developing software in safety critical systems. In recent years, Ada development has been experiencing a resurgence in popularity due to the growth of the number of safety critical systems being developed," said William McCaffrey, Chief Operating Officer for Vector Software. "Vector Software has always been strongly committed to the Ada market and our continued support demonstrates our commitment to organizations developing safety critical applications in Ada."

VectorCAST/Ada fully supports Ada 83, Ada 95, and Ada 2005. Vector Software support for Ada is available to all VectorCAST/Ada customers under standard software maintenance agreement. Additionally, the VectorCAST platform has integrations to IBM® Rational® Rhapsody®, AdaCore GNAT, Green Hills® AdaMULTI™, Atego™ ObjectAda®Aonix, and DDC-I™ SCORE®.

[See also "Vector Software — VectorCAST 5.3", AUJ 32-4 (December 2011), p. 141. —sparre]

Aonix ObjectAda 8.5 for Windows

Subject: Atego Releases Aonix ObjectAda 8.5 for Windows
Date: September 11, 2012
From: Atego Press Releases
URL: http://www.atego.com/pressreleases/pressitem/atego-launches-aonix-objectada-85

Atego™, the leading independent supplier of industrial-grade, collaborative development tools for engineering complex, mission- and safety-critical architectures, systems, software and hardware, has launched Aonix ObjectAda® 8.5 for Windows with new support for Microsoft Windows® 7 systems.

The new release is based on the latest build tools and libraries from the Microsoft Windows SDK 7.1 and Visual Studio 2010 SP1 and provides a new streamlined, modernized installer based on the Microsoft Windows Installer. Advancements to the Aonix ObjectAda debugger, Integrated Development Environment (IDE), and AonixADT support debugging of Ada code in DLLs and allow the debugger to attach to running processes. Aonix ObjectAda 8.5 includes improved usability in the documentation and online help for Windows 7. And, the product supplies the first implementation of AonixADT on Windows that supports the latest version of Eclipse a multi-language software development environment comprising an open-source (IDE) and an extensible plug-in system.

“Aonix ObjectAda for Windows is one of the most popular Ada environments of all time and is used by the majority of Atego’s ObjectAda customers for development of large-scale mission-critical applications. This release strengthens Atego’s leadership position in Ada development tools for the Windows platform,” stated Hedley Apperly, Atego’s Vice-President of Product & Marketing. “We are committed to providing support for modern, up-to-date engineering platforms such as provided by Microsoft and other hardware vendors in support of our customer requirements and demands.”

Shipping and Availability

Aonix ObjectAda 8.5 for Windows is fully released and immediately available. Product license pricing is available on request.

SPARK Pro 11

From: AdaCore Press Center
Date: November 29, 2012
Subject: AdaCore and Altran Praxis Release SPARK Pro 11
URL: http://www.adacore.com/press/spark-pro-11/

Increased verification efficiency for high-assurance systems

BATH, NEW YORK and PARIS, November 29, 2012 – High Assurance Software Symposium and SPARK User Group - AdaCore and Altran Praxis today announced the release of the SPARK Pro 11 software development and verification environment, providing a major step forward for the developers of high-assurance systems. SPARK Pro 11 offers many enhancements particularly in the area of program proof.

Major improvements to proof functions

A number of significant enhancements have been made to the way that functions and proof functions are handled in SPARK Pro 11. These changes will improve project efficiency by eliminating the vast majority of rules that were previously manually encoded. The main changes include a more powerful language for specifying proof functions and the ability to use the functions in any proof context. This greatly simplifies the task of writing and maintaining functional contracts for critical software, providing high-assurance at lower cost.

Counter-example generation

Proof is a very powerful technique for achieving high levels of assurance in safety or security-critical software. However, when performing proofs users typically spend much of their time inspecting undischarged “verification conditions” to determine whether they can indeed be proved. Included with SPARK Pro 11, Riposte is a new tool that not only determines whether a verification condition is false, but can also generate a counter-example to demonstrate the conditions under which it is false. Riposte is a major improvement to the verification workflow, saving projects a significant amount of time previously spent analyzing unprovable verification conditions and providing developers with intuitive explanations. Riposte was developed jointly by Altran Praxis and the University of Bath (UK).

Clearly defined assumptions

The new assume contract in SPARK Pro 11 allows users to introduce system-level assumptions about programs into their proofs in a clear and concise format. Previously, these assumptions might have been captured by user rules or manual review.

Availability

SPARK Pro 11 is available now. For more information please visit http://www.adacore.com/home/products/sparkpro/ or contact info@adacore.com.

Demonstration

A demonstration providing an introduction to the new features in SPARK Pro 11 is available now at http://adaco.re/2h

[See also "AdaCore / Altran Praxis — SPARK Pro 10", AUJ 32-3 (September 2011), p. 137. —sparre]

Ada and GNU/Linux

GNADE ODBC on 64-bit Debian

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 3 Sep 2012 21:17:36 +0200
Subject: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

What is the status of the bug reported by Ludovic Brenta [1].

I am asking because it seems there that the latest version of libgnadeodbc2 has the issue not resolved. In particular gnu-db-sqcli.adb has GetData declared as:

```
function GetData (
  StatementHandle : in SQLHSTMT;
  ColumnNumber : in SQL_Column_Number;
  TargetType: in SQL_C_DATA_TYPE;
  TargetValue: in SQLPOINTER;
  BufferLength: in SQLINTEGER;
  StrLen_Or_Ind: access SQLINTEGER)
  return SQLRETURN;
```

This is wrong and causes programs using GNADE ODBC crash.

BufferLength must have been SQLLEN. StrLen_Or_Ind must have been access SQLLEN. The type SQLLEN is 64-bit under Debian 64-bit (32-bit under Debian 32-bit).

[1] https://sourceforge.net/tracker/?func=detail&aid=1591916&group_id=23045&atid=377331

From: Graham Stark <graham.stark@virtual-worlds.biz>
Date: Tue, 4 Sep 2012 02:40:32 -0700
Subject: Re: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

[…]

If you just change SQLUINTEGER to SQLUBIGINT in the indicated places Ludovic's warning goes away. It seems harmless either way, to the limited extent that I understand the code - the thing that's declared as a pointer is really just a bitmap, so it's just that the top 32 bits can't be set by that function.

The SQLLEN thing looks like a different problem and really quite nasty - I hadn't come across this before. My reading of it is that some ODBC drivers might need 32 bits there even on 64 bit systems; see [1], for instance.

I have notice some funny behaviour retrieving strings from databases, where the length is not returned correctly, so maybe that's the cause. Never seen a crash, though.

[1] http://www.martin-evans.me.uk/node/99

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 4 Sep 2012 13:54:14 +0200
Subject: Re: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

[…]

Should I report it independently to sourceforge?

[…]

Specifically to unixodbc and Debian, the macro BUILD_LEGACY_64_BIT_MODE if defined in sqltypes.h, makes SQLLEN 32-bit. When undefined it is 64-bit. The effect of SQLINTEGER (always 32-bit) passed instead of 64-bit is memory corruption.

It should also become an issue for native Windows 64-bit applications, once GNAT to support them.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 07 Sep 2012 05:51:09 -0400
Subject: Re: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

As far as I know, there is no current Debian maintainer for GNADE. I was the maintainer, but I've resigned, partly because the code is hard to maintain in the first place, and needs to be re-written.

I suggest you switch to GNATcoll, which includes Ada front-ends for postgres and SQLite3 (but not ODBC). That code is better written, and actively maintained, supported by paying customers.

It also includes higher-level object oriented front-ends.

If paying customers lobby for ODBC support, it will happen.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 7 Sep 2012 03:57:37 -0700
Subject: Re: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

> […]

But GNATcoll doesn't have a Debian maintainer either and is not even packaged in Debian. If someone here on comp.lang.ada would like to make a difference, the one thing to know is that the sources of GNATcoll are part of the sources of GPS, therefore packaging GNATcoll for Debian requires modifying the packaging of gnat-gps; see [1]. As much as I would like to do that work myself, I lack time for it.

PS. AFAIU, part of GNATcoll is built on top of GNADE (and includes a fork of GNADE).

[1] http://bugs.debian.org/640532

From: Graham Stark <graham.stark@virtual-worlds.biz>
Date: Fri, 7 Sep 2012 04:25:32 -0700
Subject: Re: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

I'd think Gnade is worth persevering with.

The changes that need to be made to support 64 bit ODBC are detailed here:

http://support.microsoft.com/kb/298678

The scary bit is that (if I've read this right) sometimes function returns are 32 bit and sometimes 64 bit depending on the inputs. We're going to need a slightly thicker binding than at present to handle this, I'd think.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 7 Sep 2012 14:10:27 +0200
Subject: Re: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

[…]

Yes. Simple components provide a safer layer on top GNADE ODBC. E.g. controlled types for handles, typed SQLGetData, etc.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Fri, 07 Sep 2012 18:06:25 +0200
Subject: Re: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

GNATcoll relies a few GNAT specific packages; any experience whether these work with other Ada compilers?

with GNAT.Strings, GNATCOLL.*, GNAT.Calendar… etc. Nothing terrible, and I am not complaining, just mentioning.

Also, for those who might need copy protection for their sources, the license (GPL) might not be what they'd like, IINM.

From: Graham Stark <graham.stark@virtual-worlds.biz>
Date: Mon, 10 Sep 2012 11:10:47 -0700
Subject: Re: Status of GNADE ODBC under Debian 64-bit
Newsgroups: comp.lang.ada

It would be interesting to know if the other database interfaces out there (I know of GNATcoll and SOCI) are immune to these 64/32 bit problems.

QtAda in Debian

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 18 Sep 2012 00:16:38 -0700
Subject: Re: QtAda in Debian (was: Re: Qt or Gtk? (from an Ada perspective))
Newsgroups: comp.lang.ada

>> GtkAda is packaged in Debian, Qt is not.

> BTW: Are there any plans to do this?

Not that I know of. As usual, if someone volunteers to do that, they will find help and support on debian-ada@lists.debian.org and they will become rich and famous :)

In theory, whenever someone plans to make a new package for Debian, they open a "bug" report to say so. Look for "Ada" on the (long!) list of "Work-Needing and Prospective Packages" at http://bugs.debian.org/wnpp to get an idea of what's planned.

GPS tips of the day

From: Ada novice <shai.lesh@gmx.com>
Date: Sun, 23 Sep 2012 06:14:59 -0700
Subject: Packaged Debian Ada doesn't show tips of the day

Newsgroups: comp.lang.ada

Following an earlier discussion with Ludovic Brenta, I have put back the packaged Debian Ada on my Crunchbang Linux. What I am writing below applies to Linux Mint as well. Both Linux Mint and Crunchbang are direct derivatives of Debian. Crunchbang is so close to Debian that even its installation set up is identical to Debian!

I installed GNAT via the synaptic package manager. I think it was GNAT-4.6 and GPS alongside. On typing gnat-gps at the command line, I get in the message panel of GPS:

Welcome to GPS 5.0-12 (Debian) hosted on a i486-linux gnu)

Some year here -2010 AdaCore

Traceback (most recent call last):

File "/usr/share/gps/plug-ins /tip_of_the_day.py", line 628, in on_gps_started

Messages = GPS.MDI-get("Messages"). pywidget()

AttributeError: MDIwindow instance has no attribute 'pywidget'

It could be something simple missing that comes with Debian but not on its derivatives that is causing me to get this message. However, I do not get the above message with GNAT GPL from AdaCore.

From: Brian Drummond <brian@shapes.demon.co.uk>
Date: Sun, 23 Sep 2012 14:04:00 +0000
Subject: Re: Packaged Debian Ada doesn't show tips of the day
Newsgroups: comp.lang.ada

> […]

If I google "debian what provides pywidget" the first hit I see is [1] which looks relevant.

Looks like it might be worth installing or updating PyGTK.

[1] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=654339

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sun, 23 Sep 2012 22:08:57 +0200
Subject: Re: Packaged Debian Ada doesn't show tips of the day
Newsgroups: comp.lang.ada

> Looks like it might be worth installing or updating PyGTK.

Specifically, install the package python-gtk2 which gnat-gps recommends.

From: Ada novice <shai.lesh@gmx.com>
Date: Thu, 27 Sep 2012 22:37:26 -0700
Subject: Re: Packaged Debian Ada doesn't show tips of the day
Newsgroups: comp.lang.ada

> […]

python-gtk2 was already installed (perhaps before or was installed during the installation of gnat-gps) and I have the latest version 2.24.0-3.

I will stick to GNAT GPL which works fine.

Ada and Mac OS X

GCC 4.7.0 for Mac OS X Lion

From: Simon Wright <simon@pushface.org>
Date: Sun, 23 Sep 2012 19:58:33 +0100
Subject: GCC 4.7.0 for Mac OS X Lion
Newsgroups: comp.lang.ada

Following up on a recent posting:

https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/4.7.0/

The README says:

This is GCC 4.7.0 built for Mac OS X Lion.

Includes AUnit, GPRbuild, XMLAda from GNAT GPL 2012.

Compilers included: Ada C C++ Fortran.

Target: x86_64-apple-darwin11

```
Configured with: \
 ../gcc-4.7.0/configure \
 --prefix=/opt/gcc-4.7.0 \
 --disable-bootstrap \
 --disable-multilib \
 --enable-languages=c,ada,fortran,c++ \
 --build=x86_64-apple-darwin11
Thread model: posix
gcc version 4.7.0 (GCC)
```

Install by

```
$ cd /
$ sudo tar jxvf ~/Downloads/gcc-4.7.0-x86_64-apple-darwin11.tar.bz2
```

(and I see that it ought to add, "Ensure that /opt/gcc-4.7.0/bin is at the start of your PATH"

Installing GNAT 2012 on Mountain Lion

From: alexmartin715@gmail.com
Date: Tue, 23 Oct 2012 06:03:28 -0700
Subject: Installing GNAT 2012 on Mountain Lion
Newsgroups: comp.lang.ada

I don't know if I made a mistake during installation but I am trying to get GNAT working on Mountain Lion. It has installed but when I run gnatmake I get this:

```
gnatbind -x extra.ali
gnatlink extra.ali
ld: in /usr/bin/GNAT/lib/gcc/x86_64-apple-darwin10.8.0/4.5.4/adalib/libgnat.a(raise-gcc.o), corrupt archive, member contents extends past end of file for architecture x86_64
collect2: ld returned 1 exit status
gnatlink: error when calling /usr/bin/gnat/bin/gcc
gnatmake: **- link failed.
```

I have it installed on a current generation MBP 13 inch I7.

From: Simon Wright <simon@pushface.org>
Date: Tue, 23 Oct 2012 16:02:07 +0100
Subject: Re: Installing GNAT 2012 on Mountain Lion
Newsgroups: comp.lang.ada

I have it installed on a current gen MBP 13-inch I5, and no such problems. This is an install that was done on the previous early-2008 MBP 15-inch running - well, I can't remember when I installed Lion, so it could have been Lion or Snow Leopard - and recovered from a Time Machine backup.

I seem to have downloaded on 7 July, and the libgnat.a info is

```
$ ls -l /opt/gnat-gpl-2012/lib/gcc/x86_64-apple-darwin10.8.0/4.5.4/adalib/libgnat.a
-rw-r--r--@ 1 simon1 simon1 5764464 14 May 15:10 /opt/gnat-gpl-2012/lib/gcc/x86_64-apple-darwin10.8.0/4.5.4/adalib/libgnat.a
$ md5 /opt/gnat-gpl-2012/lib/gcc/x86_64-apple-darwin10.8.0/4.5.4/adalib/libgnat.a
MD5 (/opt/gnat-gpl-2012/lib/gcc/x86_64-apple-darwin10.8.0/4.5.4/adalib/libgnat.a) = 19dd744947c6fa1d180dfb520f6b077d
```

I can only suggest reinstalling.

Did it really install in /usr/bin/gnat/? Here, it wants to install in /usr/local/gnat/ which is much more normal. You'll see from the above that I have my own views on where to install GNAT (because I need to have several version on-the-go).

References to Publications

Ada gets a makeover

From: Jack G. Ganssle <jack@gannsle.com>
Date: September 10, 2012
Subject: Ada gets a makeover
URL: http://www.embedded.com/electronics-blogs/break-points/4395816/Ada-2012

Ada 2012 includes some important improvements.

Dan Saks noted recently that embedded.com surveys over the last dozen years indicate that we primarily use C in embedded applications. C++ is a distant second, with everything else being in the noise.

[…]

There is a small community of Ada users who routinely crank out code an order of magnitude less buggy than common in the C/C++ world. The language has been revised several times. The latest is Ada 2012 which is expected to get an ISO approval this year.

The new version has some fantastic additions.

[…]

What does it mean to say that code must work all the time?

From: Robert Dewar
Date: September 27, 2012 11:56 AM
Subject: What does it mean to say that code must work all the time?
URL: http://electronicdesign.com/article/embedded/code-work-time-74483

[About tolerating defects in software. —sparre]

From: Steve Morton
Date: 2012-10-01
Subject: Re: What does it mean to say that code must work all the time?
Source: LinkedIn

It's important to remember that process is a means to achieve correctness, as opposed to the measure of correctness. Sometimes it seems that we have forgotten that we put process steps in place to avoid repeating past mistakes (our own or those committed by others).

No programmer can anticipate every possible combination of conditions, to explicitly address every situation that could conceivably arise. But having a fail-safe state, where unknown issues cause the system to revert to a controllable status to allow for a safe conclusion of a given flight (or sequence in an industrial setting) is very important. The difficulty lies, in my experience, in precluding a corner case from slipping beneath the radar and causing the software to make just the wrong decision at just the wrong time.

As engineers, we often loathe the phrase "I don't know", yet we must continually hone our skills in making the software we write come to just that determination - and then to have it make the safest decision possible. In situations like these, technical psychology almost become art.

Ada Inside

A little software piracy

From: Brian Drummond <brian@shapes.demon.co.uk>
Date: Tue, 11 Sep 2012 17:53:44 +0000
Subject: A little software piracy
Newsgroups: comp.lang.ada

In honour of September 19, "Talk Like a Pirate Day"

http://www.dragondreamscreations.co.uk/pirate.html

With apologies to the CBSG project authors…

SecureOne™

From: AdaCore Press Center
Date: September 17, 2012
Subject: Rockwell Collins Develops SecureOne™ with SPARK Pro and GNAT Pro High-Security
URL: http://www.adacore.com/press/secureone/

NEW YORK, PARIS, and CEDAR RAPIDS, Iowa, September 17, 2012 - Design East Boston 2012 - AdaCore today announced the successful usage of its SPARK Pro and GNAT Pro High-Security products by Rockwell Collins in the development of the SecureOne™ Guard, a high assurance cross domain guard for military tactical systems. The SecureOne Guard has strict requirements for reliability and security. In order to meet these needs, Rockwell Collins selected SPARK Pro and GNAT Pro High-Security as key development tools for the project.

Rockwell Collins selected the GNAT Pro High-Security and SPARK Pro development tools to support the redeployment of the cross domain guard software from the UCDMO Baseline-approved TurnstileTM Cross Domain Guard to a Multiple Independent Levels of Security (MILS) Real Time Operating System (RTOS) as part of the SecureOne project development. The SecureOne Guard is one of the five SecureOne cross domain technologies for high assurance military tactical systems.

“Software at the highest security assurance levels needs to be developed with the most trustworthy languages and tools,” said Robert Dewar, AdaCore President and CEO. “The SPARK programming language, and its supporting toolset, meets these requirements, allowing formal demonstration of security-related properties, such as absence of run-time exceptions. We are pleased that Rockwell Collins, a longstanding AdaCore customer, chose our SPARK Pro tools and GNAT Pro High-Security technology to develop the SecureOne components.”

[…]

About SecureOne

SecureOne is a family of high assurance cross domain technologies, providing trusted multi-classification information sharing for military tactical systems. Using SecureOne, users can securely access both unclassified and classified data on the same equipment while benefiting from reduced size, weight and power.

The GNU Ada KDF9 emulator, now on Raspberry Pi

From: Bill Findlay <yaldnif.w@blueyonder.co.uk>
Date: Wed, 03 Oct 2012 07:26:15 +0100
Subject: ee9, the GNU Ada KDF9 emulator, now on Raspberry Pi
Newsgroups: comp.lang.ada

ee9, my GNU Ada emulator of the English Electric KDF9, has been available for some time: for the x86_64 and PowerPC G5 architectures under Mac OS X; for the x86_32 and x86_64 architectures under Linux/FreeBSD; and for the x86_32 architecture under Microsoft Windows (XP/SP3 or newer).

It has now also been implemented for the ARM architecture, in the form of the Raspberry Pi educational microcomputer running under Raspbian Linux (i.e. Debian Wheezy for ARM11).

Download packages for all of these hosts, including executable binaries and complete sources, can be found at: <http://www.findlayw.plus.com/KDF9/#Emulator>

ee9 is distributed as free software under the terms of GNU General Public License.

[See also "KDF9 emulator in Ada 2005", AUJ 32-2 (June 2011), p. 74. —sparre]

ADHCP

From: Adrian-Ken Rueegsegger <ken@codelabs.ch>
Date: Thu, 15 Nov 2012 15:38:39 +0100
Subject: ANN: ADHCP version 0.3
Newsgroups: comp.lang.ada

I am proud to announce the first official release of ADHCP, which is an implementation of the DHCP protocol in Ada. Currently the project provides client and relay services for DHCPv4.

The project website is at [1], the current release 0.3 can be downloaded from [2].

The ADHCP DHCPv4 client (adhcp_client) is D-Bus aware and can be used on most modern Linux distributions as a drop-in replacement for other clients such as ISC's dhclient or busybox's udhcpc. Instructions on how to use adhcp_client on a Linux desktop are available on the project website.

adhcp_relay is a DHCPv4 and BOOTP relay agent written in Ada. The relay agent listens for DHCPv4 or BOOTP queries from clients or other relay agents on a given interface, forwarding them to a specified upstream server or relay agent.

The ADHCP implementation is designed to be simple and supports only essential features while still conforming to the related DHCP RFCs. A small text file documenting the RFC compliance of ADHCP is part of the project documentation [3].

[1] - http://www.codelabs.ch/adhcp/

[2] - http://www.codelabs.ch/download/

[3] - http://www.codelabs.ch/adhcp/rfc-conformity.html

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Sun, 18 Nov 2012 13:00:21 +0100
Subject: Re: ADHCP version 0.3
Newsgroups: comp.lang.ada

Randy Brukardt wrote:

> In this specific case, I'm not sure what use a DHCP client would be useful for - getting IP addresses is something that has to be done very early in a system's life

Or whenever you move it to a different network.

> and thus belongs to the kernel/OS and not the application program.

On unix systems the DHCP client is typically running in user-space, and as such a part of the OS you can switch for a different implementation practically as easily as any application.

I think replacing unsafe/buggy parts of an OS with safer/less buggy implementations is a worthy goal. And if using Ada can help getting there, it is both fine and a good way to promote our favourite programming language.

> I could see if having use in an all-Ada system but that's about it.

Once it comes as a package in Debian, I expect to use instead of the ISC DHCP client I use at the moment.

Java Virtual Machine

From: Frederic BOYER <frederic.boyer.kx@gmail.com>
Date: November 23, 2012
Subject: jadam
URL: http://sourceforge.net/projects/jadam/

Jadam is an effort to code a Java Virtual Machine using the Ada 2005 language.

AZip

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Sun, 25 Nov 2012 08:53:05 -0800
Subject: Ann: AZip 0.95
Newsgroups: comp.lang.ada

AZip [1] is a Zip archive manager.

Some features:

- multi-document (should be familiar to MS Office users)
- simple to use (or at least I hope so ;-)) - portable (currently on GWindows, but the key parts of the UI and user persistence are generic)
- uses the highly portable Zip-Ada library
- useful tools:
  - integrity check
  - text search function through archive, without extracting files
  - (*) archive refresher
  - (*) archive comparison
  - (*) archive recompression
  - (*) archive merge

AZip is from A to Z in Ada :-)

(*): not yet implemented but some exist as command line tools with Zip-Ada

[1] http://sf.net/projects/azip/

Ada in Context

Identifier casing considered ugly

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Wed, 05 Sep 2012 21:01:02 +0200
Subject: Ada's identifiers casing considered ugly
Newsgroups: comp.lang.ada

[…] http://www.python.org/dev/peps/pep-0008/

> The following naming styles are commonly distinguished:
>
> […]
>
> Capitalized_Words_With_ Underscores (ugly!)

With an exclamation mark, above all.

Just to get back to a more interesting topic, I also had another though while reading this PEP: this may help if there was such a precise layout convention for Ada as the one there is for Python. But may be the one of AdaCore is more widespread than I believe (we already discussed this a bit with in old topic about the *_Type convention).

From: Pascal Obry <pascal@obry.net>
Date: Wed, 05 Sep 2012 21:23:50 +0200
Subject: Re: Ada's identifiers casing considered ugly
Newsgroups: comp.lang.ada

>> Capitalized_Words_With_ Underscores (ugly!)

iunderstandthemasitisreallyhardtoreadatextwithunderscoresseparatingwords.

From: Vinzent Hoefler <0439279208b62c95f1880bf0f8776eeb @t-domaingrabbing.de>
Date: Wed, 05 Sep 2012 21:37:42 +0200
Subject: Re: Ada's identifiers casing considered ugly
Newsgroups: comp.lang.ada

>>> Capitalized_Words_With_ Underscores (ugly!)

> iunderstandthemasitisreallyhardtoreadatextwithunderscoresseparatingwords.

AndItSureDoesNotLookSoUgly.

From: Manuel Collado <m.collado@domain.invalid>
Date: Wed, 05 Sep 2012 23:07:25 +0200
Subject: Re: Ada's identifiers casing considered ugly
Newsgroups: comp.lang.ada

> […]

Ada code is case insensitive. So a code reformatter can safely change the case of identifiers to a normalized form, in order to avoid variants of the same identifier inside a set of source files. If there are no underscores to delimit components of a composed identifier, then they can/should be merged in a single word. Underscores as separators ensure that the composed name remains a composed one.

After that, capitalization of individual components of a composed identifier remains a matter of taste. Some people will find it easier to read (w.r.t. all lowercase or uppercase).

From: Vinzent Hoefler <0439279208b62c95f1880bf0f8776eeb @t-domaingrabbing.de>
Date: Wed, 05 Sep 2012 21:15:49 +0200
Subject: Re: Ada's identifiers casing considered ugly
Newsgroups: comp.lang.ada

> […] a precise layout convention for Ada […]

What's wrong with the famous "Ada 95 Quality & Style: Guidelines for Professional Programmers"? ;)

From: Nasser M. Abbasi <nma@12000.org>
Date: Wed, 05 Sep 2012 18:23:13 -0500
Subject: Re: Ada's identifiers casing considered ugly
Newsgroups: comp.lang.ada

>> Capitalized_Words_With_ Underscores (ugly!)

I agree it is ugly. I said this same thing myself many times.

http://coding.derkeiler.com/Archive/Ada/comp.lang.ada/2012-08/msg00219.html

“I like underscore in names of variables. But I do not like at all the current Ada tradition of using Upper case for first letter as in

This_Is_An_Identifier

I find this ugly and hard on the eye to read. I like all lower case for variable names, as it is easier to read. Less variation in texture and form

this_is_an_identifier”

So, I am not the only one then who thinks this Ada naming style is ugly ;)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 16 Oct 2012 18:07:32 -0500

Subject: Re: Ada's identifiers casing considered ugly
Newsgroups: comp.lang.ada

> This_Is_An_Identifier

> […] ugly and hard on the eye to read.

I agree with this.

> [easier to read]

> this_is_an_identifier

But I don't agree with this. This blends identifiers and reserved words into an unreadable mess. (Maybe it's OK if you ONLY use colorizing editors and never expect anyone else to use your code.)

RRS uses "title case" for identifiers, which means that we use the same casing as you would if you wrote this as the title of something. For the above, that gives:

This_is_an_Identifier

Our pretty printer can enforce this style directly (it has a list of words that aren't capitalized).

I've also experimented with capitalizing just the first letter:

This_is_an_identifier

I definitely agree that capitalizing words like "Is" is ugly. I get in trouble repeatedly working on the Ada Standard, because I naturally refuse to type "_Is_" and thus a lot of identifiers end up in the "wrong" case.

Tasking and timing out

From: Thomas Løcke <tl@ada-dk.org>
Date: Thu, 20 Sep 2012 10:17:22 +0200
Subject: Tasking and timing out
Newsgroups: comp.lang.ada

I need to be able to be abandon a socket connection in a task and let the task complete if:

a. the socket connect takes too long to complete or

b. the Start task entry isn't called within a set timeframe

My current solution appears to do the trick just fine, but I'm not sure if it's a "good" solution, specifically the nested select block pains me to no end - I can't explain why though. :)

```
task body AMI_Action is
begin
  select
    accept Start do
      select
        delay 2.0;
        raise AMI_Action_Error with
          Connect_Timed_Out_Message;
      then abort
        AWS.Net.Std.Connect (
         Socket => Action_Socket,
          Host   => Config.Get (PBX_Host),
          Port   => Config.Get (PBX_Port));
      end select;
    end Start;
  or
    delay 3.0;
    raise AMI_Action_Error with
            No_Call_On_Start_Received;
  end select;

  loop
    -- Mad socket action!
  end loop;
exception
  -- Catch exceptions, log issue and
  -- complete.
end AMI_Action;
```

The general idea is that if the call to AWS.Net.Std.Connect takes more than 2 seconds to complete, then I raise an exception or if the Start entry isn't called before 3 seconds have passed, then I also raise an exception. In both cases I just want the task to log the issue and complete.

It works - at least I haven't been able to make it fail in my tests, it's just that I'm not sure if this is the right way to live up to the requirements - maybe there's a better way?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 20 Sep 2012 10:53:42 +0200
Subject: Re: Tasking and timing out
Newsgroups: comp.lang.ada

> I need to be able to be abandon a socket connection in a task and let the task complete if:

ATC is almost never a good idea. I wonder how it worked in your case. Maybe because the default socket timeout was incidentally close to 2 sec?

In any case you should not expect Ada run-time aborting outstanding OS I/O requests. In some better world, but not under Windows or Linux.

A proper solution for blocking sockets is to close the socket from an independent task and catching socket error when it propagates or else to manipulate the socket timeout before starting any blocking operation.

From: Thomas Løcke <tl@ada-dk.org>
Date: Thu, 20 Sep 2012 11:36:38 +0200
Subject: Re: Tasking and timing out
Newsgroups: comp.lang.ada

> ATC is almost never a good idea. I wonder how it worked in your case. Maybe because the default socket timeout was incidentally close to 2 sec?

I tested this with a call to a dummy procedure that did nothing but hang for X seconds, so the fact that it works have nothing to do with the default socket timeout. When X > 2 the exception was raised, and when X < 2 then the task proceeded to enter the loop.

ATC might almost never be a good idea, but that "almost" surely means that in some cases it can be a good idea? :)

As it is said in the John Barnes book "Programming in Ada 2005":

"The general effect can of course be programmed by the introduction of an agent task and the use of the abort statement but this is a heavy solution not at all appropriate for most applications needing a mode change."

The "then abort" syntax seems appropriate for this specific case, where all I want is a completed task in case things aren't happening within a set time limit.

> In any case you should not expect Ada run-time aborting outstanding OS I/O requests. In some better world, but not under Windows or Linux.

I had not thought about that. What you're saying is that despite the "then abort" syntax I cannot safely rely on the termination of the ongoing AWS.Net.Std.Connect procedure? In that case then this is indeed a rather nasty problem. Again from John Barnes:

"The general idea is that if the statements between "then abort" and "end select" do not complete before the expiry of the delay then they are abandoned and the statements following the delay are executed instead."

If I cannot rely on the runtime to actually abandon the call to AWS.Net.Std.Connect, then I wonder what circumstances John Barnes is referring to?

> A proper solution for blocking sockets is to close the socket from an independent task and catching socket error when it propagates or else to manipulate the socket timeout before starting any blocking operation.

I did think about having an agent task to manage this, but it seemed messy and heavy compared to the extreme simplicity of this solution. Also I found it hard to meet my two requirements.

I felt relatively safe using "then abort" because John Barnes specifically mentions this form of select statement as a solution to the problem of quitting action A and do B instead if A doesn't complete within a specified duration.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 20 Sep 2012 12:21:19 +0200
Subject: Re: Tasking and timing out
Newsgroups: comp.lang.ada

ARM 9.8 provides the guideline.

Specifically, you should consider:

1. Operations which can be prematurely aborted (no side effects)

2. Operations which cannot be aborted:

  2.1 Ones for which 9.8 provides safety against abort:

    2.1.1. Ones which are time bounded

    2.1.2. Ones which could deadlock when abort attempted (difficult to construct, but I would not exclude a possibility of)

2.2 Ones for which 9.8 does not provide cover:

2.2.1. Ones for which abort would cause malfunction (memory, resource leaks, corruption etc)

2.2.2. Ones of which premature termination is impossible, e.g. system calls, most I/O, waitable system objects etc.

> […]

There is not much difference between ATC and explicit aborting tasks. Both represent the case when the task control flow is preempted. The perils are basically same.

> I had not thought about that. What you're saying is that despite the "then abort" syntax I cannot safely rely on the termination of the ongoing AWS.Net.Std.Connect procedure?

Not if that is 1) an OS-native thread, 2) blocked by the OS, e.g. upon waiting for an outstanding I/O request.

A good example is using ATC when reading the keyboard input using Get_Line. Depending on the implementation of Get_Line it may work or may not work. 80 to 20 for the latter.

> If I cannot rely on the runtime to actually abandon the call to AWS.Net.Std.Connect, then I wonder what circumstances John Barnes is referring to?

#1 + #2.1.1 above. E.g. some computations, even with controlled types involved. Anything else requires *careful-analysis and thus is necessarily very fragile. Ergo, for the software design POV, it is better to forget about ATC, too problematic.

> I did think about having an agent task to manage this, but it seemed messy and heavy compared to the extreme simplicity of this solution.

Not really. Usually you have some worker task anyway due to the asynchronous nature of I/O and because you liked to share the connection between several tasks.

The clients are separate tasks asking the worker for data ready. They would do that by a timed entry call to the worker's entry or else to a protected object's entry. Upon timeout of this call you would call some Cancel operation on the communication object which internally would close the socket at some stage. That in turn will cause an error in the worker task, which would drop the connection and then would try to reconnect etc

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 20 Sep 2012 11:33:46 -0700
Subject: Re: Tasking and timing out
Newsgroups: comp.lang.ada

First of all, doesn't AWS.Net.Std.Connect have its own facility for handling timeouts? I'm looking at the source, and it looks like it does, but I don't know if I'm looking at the most recent version. If it does, you should definitely use that, and not rely on ATC.

> I had not thought about that. What you're saying is that despite the "then abort" syntax I cannot safely rely on the termination of the ongoing AWS.Net.Std.Connect procedure?

That's correct. Here are my thoughts on this, to add to what Dmitry said:

If your 2-second period times out, the call to AWS.Net.Std.Connect is *aborted*. RM 9.8 defines exactly what that means. The important thing is that it defines certain points called "abort completion points". They include things like the beginning or end of an ACCEPT statement, the beginning of end of a DELAY statement, entry calls, and a number of other cases. When the code being aborted gets to one of those points, then it will be aborted. However, the language doesn't define whether the code could be aborted *before- one of those points. In particular, a blocking I/O operation (such as Ada.Text_IO.Get_Line from a console, or a socket connect) is *not- an abort completion point. It's up to an Ada implementation to decide where things could be aborted. In your case, since you're calling a procedure in another package, you'd have to peek at the source of that package to find out where abort completion points might be happening.

If the code in Connect is executing some OS operation that's doing the connect, then aborting the code could have several different effects, depending on the Ada implementation:

(1) The OS operation is not affected, since it's not an abort completion point. The program still waits for the OS call to return; and then, later, when some other abort completion point is reached, the abort will finally take place.

(2) The expiration of the "delay 2.0" causes some sort of interrupt handler to be executed. This may or may not disrupt the OS call. The interrupt handler then manipulates the program stack to cause it to continue executing at the statement after the "delay 2.0". Whether this actually works or not is highly OS- and implementation-dependent, and it's entirely possible that on some OS's it could do damage.

(3) The Ada implementation provides facilities for abortable I/O operations, and integrates this with its runtime so that an abort will work properly. Of course, this means that AWS.Net.Std.Connect would have to use those facilities, rather than calling the OS operation directly. Also, a facility like this would have to be written separately for each kind of I/O operation.

Whether a facility *could- be written to make a particular type of I/O abortable depends on the OS, and I don't think it can always be done. On Linux, there's a connect() call to connect to a socket. But I don't think there's a cancel_connect() call that would let you terminate a connect() call that had started (perhaps in a different thread). This means that if you want a timeout, you probably have to know *ahead- of time, before you call connect(), whether the connect() will have an expiration period and how long it is. (And I'm not even 100% sure that you can do a "timed connect" in Linux even if you do know.)

From: Thomas Løcke <tl@ada-dk.org>
Date: Thu, 20 Sep 2012 22:40:57 +0200
Subject: Re: Tasking and timing out
Newsgroups: comp.lang.ada

> Ergo, for the software design POV, it is better to forget about ATC, too problematic.

I've taken your advice to heart Dmitry, and re-coded the routine without ATC, and it is much better for it.

From: Thomas Løcke <tl@ada-dk.org>
Date: Thu, 20 Sep 2012 22:43:41 +0200
Subject: Re: Tasking and timing out
Newsgroups: comp.lang.ada

> […]

Awesome explanation Adam! Your post really drove the point Dmitry made home.

And yes, AWS does have some facilities for this, and yes I'm using those in my new non-ATC solution.

"then abort" is out the window!

Optimization away of checks in 'Valid

From: Joseph Wisniewski <wisniewski.ru@gmail.com>
Date: Tue, 25 Sep 2012 18:33:54 -0700
Subject: optimization away of checks in 'valid
Newsgroups: comp.lang.ada

Ran into an issue with one compiler having to do with the implementation of 'Valid. Was looking for comments as to how other compilers handle this.

Basically, the question is, if 'Valid is called on an integer object, are there conditions under which some of the checks done by 'Valid (range checking on an object of an integer subtype) are removed?

Specifically, we had a case where C++ code was not checking the bounds of an integer subtype as it was passed to Ada code via a function parameter. The Ada code _was_ checking via 'Valid. 'Valid returned true even though the integer value was out of bounds. Turns out the compiler relied on the "allowed assumption" that all callers "check their bounds" for such data. As such, the range checks in 'Valid were eliminated as redundant as part of building with optimization on. In fact, I believe the checks were eliminated under no-opt also.

My question is whether this is similar behavior across compilers or if 'Valid is viewed as always performing the same checks, including and perhaps especially bounds checks in this kind of situation.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 26 Sep 2012 01:35:24 -0700
Subject: Re: optimization away of checks in 'valid
Newsgroups: comp.lang.ada

> Turns out the compiler relied on the "allowed assumption" that all callers "check their bounds" for such data.

I'd think this assumption should be disallowed for exported subprograms, or actually for all subprograms with a Convention other than Ada?

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Wed, 26 Sep 2012 14:01:14 +0200
Subject: Re: optimization away of checks in 'valid
Newsgroups: comp.lang.ada

> Specifically, we had a case where C++ code was not checking the bounds of a integer subtype as it was passed to Ada code via a function parameter. The Ada code _was_ checking via 'valid. 'valid returned true even though the integer value was out of bounds. Turns out the compiler relied on the "allowed assumption" that all callers "check their bounds" for such data.

FWIW, is the function like any of the ones below? I see Exported_2 and Exported_3 have 'Valid removed after compilation, as expected maybe.

```
with Interfaces.C;
package Rcheck is
  use type Interfaces.C.int;
  subtype T is Interfaces.C.int
        range -7 .. 100;
  subtype int is Interfaces.C.int;
  function Exported_1 (Item : T) return T;
  pragma Export (C, Exported_1);
  function Exported_2 (Item : int) return int;
  pragma Export (C, Exported_2);
  function Exported_3 (Item : int) return T;
  pragma Export (C, Exported_3);
  Default_Value : constant T := 33;
end Rcheck;
package body Rcheck is
  Dummy : constant := 66;
  function Exported_1 (Item : T) return T is
    Result : T;
  begin
    if Item'Valid then
          Result := Dummy;
    else
          Result := Default_Value;
    end if;
    return Result;
  end Exported_1;

  function Exported_2 (Item : int)
                              return int is
    Result : int;
  begin
    if Item'Valid then
          Result := Dummy;
    else
          Result := Default_Value;
    end if;
    return Result;
  end Exported_2;

  function Exported_3 (Item : int) return T is
    Result : T;
  begin
    if Item'Valid then
          Result := Dummy;
    else
          Result := Default_Value;
    end if;
    return Result;
  end Exported_3;
end Rcheck;
```

From: Jeffrey Carter <jrcarter@acm.org>
Date: Wed, 26 Sep 2012 09:46:42 -0700
Subject: Re: optimization away of checks in 'valid
Newsgroups: comp.lang.ada

Using any scalar subtype other than those in Interfaces.C when interfacing to C/++ is asking for trouble.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Thu, 27 Sep 2012 02:00:08 -0400
Subject: Re: optimization away of checks in 'valid
Newsgroups: comp.lang.ada

[…]

See ARM 13.9.2; it lists the operations that can return invalid values.

[…]

Reading an input parameter is not in the list of operations that can return invalid values.

"interfacing to another language" is on the list, which is what you are doing, but I suspect that really means "calling a subprogram implemented in another language"; i.e., a subprogram with pragma Import.

Do you have pragma Export on the Ada function? That would mean reading the input parameter _is_ interfacing to another language, and you could complain to your compiler vendor.

Highest bit, statically determined

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Sat, 29 Sep 2012 19:34:09 +0200
Subject: highest bit, statically determined
Newsgroups: comp.lang.ada

Is there a shorter/better way of having the compiler find the highest bit = 1 in a static numeric constant?

If N is such a constant, e.g. Some_Type'Last where Some_Type'Size = 8 and its bound are static, then

```
Highest_Bit_In_Octet : constant :=
  Natural'Max
  (8- Boolean'Pos (N / 2**7 > 0),
   Natural'Max
    (7- Boolean'Pos (N / 2**6 > 0),
     Natural'Max
     (6- Boolean'Pos (N / 2**5 > 0),
      Natural'Max
      (5- Boolean'Pos (N / 2**4 > 0),
       Natural'Max
       (4- Boolean'Pos (N / 2**3 > 0),
        Natural'Max
        (3- Boolean'Pos (N / 2**2 > 0),
         Natural'Max
         (2- Boolean'Pos (N / 2**1 > 0),
          Natural'Max
          (1- Boolean'Pos (N / 2**0 > 0),
                                   0))))))));
```

From: Pascal Obry <pascal@obry.net>
Date: Sat, 29 Sep 2012 20:11:11 +0200
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

```
if N > 128 then
  return 8;
elsif N > 64
  return 7;
elsif …

elsif N > 0 then
  return 1;
end if;
```

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Sat, 29 Sep 2012 20:59:43 +0200
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

> […]

The N > 2**X part is good, thanks for the answer that removed a fair bit of fog. But the "return X" indicates that the solution cannot be static, or can it?

There might be an expression in Ada 2012 that does it. Alas, I cannot use Ada 2012 yet -- which I should have mentioned!

```
Highest_Bit_In_Octet_2012 : constant :=
 (if    N >= 2**7 then 8
  elsif N >= 2**6 then 7
  elsif N >= 2**5 then 6
  elsif N >= 2**4 then
  elsif N >= 2**3 then 4
  elsif N >= 2**2 then 3
  elsif N >= 2**1 then 2
  elsif N >= 2**0 then 1
  else  0);
```

[with later fix merged in —sparre]

From: Bill Findlay <yaldnif.w@blueyonder.co.uk>
Date: Sat, 29 Sep 2012 19:57:08 +0100

Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

> […]

In my experience that sort of code, applied in non-static cases, is less efficient than one would hope, and more obvious code works faster.

Something like the following can be readily extended to greater operand widths:

```
function First_1_Bit (y : Octet)
return Natural is
   x : Octet;
   r : Natural;
begin
   if y = 0 then return 0; end if;
   if (y / 16) /= 0 then
      r := 4; x := y / 16;
   else
      r := 0; x := y;
   end if;
   if (x / 4) /= 0 then
      r := r + 2; x := x / 4;
   end if;
   if (x / 2) /= 0 then
      r := r + 1;
   end if;
   return r + 1;
end First_1_Bit;
```

It looks fairly inline-able, and foldable for a static value of y.

From: Bill Findlay <yaldnif.w@blueyonder.co.uk>
Date: Sat, 29 Sep 2012 20:16:01 +0100
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

> […]

I can now confirm that with GNAT GPL 2012 at -O3 it does inline and fold, but I now see that you want the result to be static as well as the operand, and this does not achieve that.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Sat, 29 Sep 2012 23:36:59 +0200
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

> function First_1_Bit

[…]

Daringly, I have tried to steal the idea and try a comparison (out of curiosity, not for the static thing). GCC performs simple tail call elimination (explaining the Shift parameter)!

```
function First_1_Bit_A (y : Octet; Shift :
         Integer := 0)  return Natural is
begin
   if y >= 2**4  then
      if y >= 2**6 then
         return Shift + 7 + Boolean'Pos
            (y >= 2**7);
      else
         return Shift + 5 + Boolean'Pos
            (y >= 2**5);
      end if;
   else
      if Y = 0 then
         return 0;
      else
         return First_1_Bit_A
            (y- 2**4, Shift => -4);
      end if;
   end if;
end First_1_Bit_A;
```

Don't know if that's as readily adaptable to other word sizes, but it might make the functionist happier. ;-)

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Sun, 30 Sep 2012 00:06:51 +0200
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

[…]

First_1_Bit is better [than First_1_Bit_A] by between 30% and 40% speed here.

Inlining can mean that the program runs about 4x as fast, for each of the two functions.

(But, with GNAT, the deprecated -gnatN has *no- effect in the case of First_1_Bit_A. Best options: -O2 -funroll-loops -gnatp -gnatn.)

From: Vadim Godunko <vgodunko@gmail.com>
Date: Sun, 30 Sep 2012 08:01:54 -0700
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

If you don't need static function, you can use following function (GCC can precompute its result for static value):

```
with Interfaces.C; use Interfaces.C;
function Bit (X : Unsigned)
         return Unsigned;
pragma Inline (Bit);
function Bit (X : Unsigned)
         return Unsigned is
   function CLZ (X : Unsigned)
                     return Unsigned;
   pragma Import (Intrinsic, CLZ,
                  "__builtin_clz");
begin
   if X = 0 then
      return 0;
   else
      return Unsigned'Size - CLZ (X);
   end if;
end Bit;
```

From: Anatoly Chernyshev <achernyshev@gmail.com>
Date: Sun, 30 Sep 2012 08:39:41 -0700
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

Ouch…

```
with Ada.Numerics.Elementary_Functions;
use Ada.Numerics.Elementary_Functions;

Highest_Bit_In_Octet := Natural
   (Float'Truncation (Log (Float (N), 2.0)));
```

I didn't check it for speed though. Pros that it doesn't depend on the size.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Mon, 01 Oct 2012 10:07:33 +0200
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

> […]

I did. And this time I had the program actually call the randomizing Initialize I had written for the test data. .-/ Results have changed in favor of the recursive first_1_bit_a. The test is running on a laptop, so it likely says little about results on a microcontroller.

x an array of 12_000 octets, 10_000 iterations for each test.

First column: Count (f(x{k}) = f(x_{k + 1})),

Second column: time in seconds

```
68580000 2.635991000
  -- f is First_1_Bit
68580000 2.036651000
  -- f is First_1_Bit_A
68580000 27.735934000
  -- f is First_1_Bit_Log
```

with inline expansion:

```
68580000 1.687923000
68580000 1.447859000
68580000 24.48147200
```

Slightly faster in both cases when translation suppresses checks, the integer versions more than the FPT version.

From: Anatoly Chernyshev <achernyshev@gmail.com>
Date: Mon, 1 Oct 2012 01:52:43 -0700
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

[…] Here's the draft lightning-fast version:

```
Last_Bit_A : array (0 .. 255) of Natural := (
         0          => 0,
         1          => 1,
         2..3       => 2,
         4..7       => 3,
         8..15      => 4,
         16..31     => 5,
         32..63     => 6,
         64..127    => 7,
         128..255 => 8);
```

[with a fix to match the reference implementation —sparre]

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Mon, 01 Oct 2012 23:30:24 +0200
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

```
68580000 2.653934000
68580000 2.021029000
68580000 27.702262000
68580000 1.173348000
-- first_1_bit_table
```

I guess the approaches can be used together for larger N-tests.

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 1 Oct 2012 15:55:43 -0700
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

I wouldn't be surprised if you could make it faster by making it a constant.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Tue, 02 Oct 2012 01:25:19 +0200
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

It is a constant. Making it variable does not make a difference, though.

From: Brian Drummond <brian@shapes.demon.co.uk>
Date: Tue, 2 Oct 2012 11:03:22 +0000
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

Just for completeness,

```
function First_Bit_Case (N : Octet)
         return natural is
begin
  case N is
    when 0      => return 0;
    when 1      => return 1;
    when 2..3    => return 2;
    when 4..7    => return 3;
    when 8..15   => return 4;
    when 16..31  => return 5;
    when 32..63  => return 6;
    when 64..127 => return 7;
    when others  => return 8;
  end case;
end First_Bit_Case;
```

Here (on an Atom netbook) it measures about the same as the equivalent if

chain; faster than the recursive versions but slower than the table.

```
--Z := Z + First_1_Bit_Ifchain(Data(i));
-- real 0m3.208s
--Z := Z + First_Bit_Table(Data(i));
-- reaL 0m1.971s
--Z := Z + First_1_Bit(Data(i));
-- real 0m4.672s
--Z := Z + First_1_Bit_A(Data(i));
-- real 0m3.937s
 Z := Z + First_Bit_Case(Data(i));
-- real 0m3.272s
```

The Ada-2012 case expression was no faster.

From: Kalvin <kalvink65@gmail.com>
Date: Wed, 3 Oct 2012 02:30:51 -0700
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

How about binary search algorithm with constant execution time

```
Binary_Search_Highest_Bit_In_Octet_2012:
constant :=
   (if N > (2**4)-1 then
      -- determine upper or lower nibble
      -- upper nibble
      if N > (2**6)-1
       -- bits 7 and 6
       if N > (2**7)-1 then
         7
       else
         6
      else
       -- bits 5 and 4
       if N > (2**5)-1 then
         5
       else
         4
    else
      -- lower nibble
      if N > (2**2)-1 then
       -- bits 3 and 2
       if N > (2**3)-1 then
         3
       else
         2
      else
       -- bits 1 and 0
       if N > (2**1)-1 then
         1
       else
         0);
```

Disclaimer: I am not an Ada programmer, so this might not compile, but describes the idea anyway.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 04 Oct 2012 09:46:16 +0200
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

> […]

For the recursive one, the above one, and the one using a table, respectively, I get, in seconds:

1.62, 1.57, 0.95

From: Tomi Saarnio <kalvin.news@gmail.com>
Date: Thu, 4 Oct 2012 03:01:50 -0700
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

> [First_Bit_Table]

This can be optimised for improved size so that we create a lookup table only for a nibble values (0..15), and divide the handling into two parts according to whether the higher nibble is zero or non-zero. The end result would be a slighter slower execution speed but less code space. And this method can easily be extended to larger data sizes, too.

From: Anatoly Chernyshev <achernyshev@gmail.com>
Date: Fri, 5 Oct 2012 00:50:15 -0700
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

If you play with the memory optimization (I'm not very good at it), the array above will be made of 4 bit integers (0 to 8), which occupy lousy 4*256/8 = 128 bytes of memory. Laughable by today's standards. Even if you go for long_long_long…_integers, the table will be quite small.

From: Anatoly Chernyshev <achernyshev@gmail.com>
Date: Fri, 5 Oct 2012 01:38:18 -0700
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

Here we go:

```
with Text_IO; use Text_IO;
procedure Table_Size is
  type Small_Integer is range 0 .. 8;
  type Small_Integer_Array is array
        (0 .. 255) of Small_Integer;
  for Small_Integer_Array'
        Component_Size use 8;
  Last_Bit_A : Small_Integer_Array :=
        0        => 0,
        1        => 1,
        2..3     => 2,
        4..7     => 3,
        8..15    => 4,
        16..31   => 5,
        32..63   => 6,
        64..127  => 7,
        128..255 => 8);
begin
  Put_Line (Integer'Image
        (Last_Bit_A'Size / 8));
end Table_Size;
```

From: Bill Findlay <yaldnif.w@blueyonder.co.uk>
Date: Sun, 04 Nov 2012 22:00:52 +0000
Subject: Re: highest bit, statically determined
Newsgroups: comp.lang.ada

> What means folding in this context?

Compile-time evaluation, so that the end result is a compile-time known (I do not say static, as that is a term of art in Ada) value.

API design considerations

From: Maciej Sobczak <maciej@msobczak.com>
Date: Mon, 8 Oct 2012 02:03:03 -0700
Subject: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

I am working on the API design where there is a need for a record type with field(s) that are arrays of some predefined type (say, Integer). So, an example record type might look like this:

```
type R is record
  X : Integer;
  Y : Unbounded_String;
  Z : Array_Of_Integers_That_We_
          Talk_About_Here;
end record;
```

The trouble is, the types involved should comply with the following requirements:

- the enclosing record type should not be limited (it should be copyable)
- the record type can be explicitly controlled, if necessary
- the record type should be definite (it should not require initialization)
- it should be possible to assign to Z, just as it is possible to assign to X and Y, and the new value for Z *might have a different length*

An important relaxation is that it is not necessary to support in-place modifications of the field in question. That is, Y above can be modified (by appending or modifying individual characters, for example) in place. It is not necessary to support it for Z, assignment of the whole array is the only operation that is needed there.

A straightforward choice is a predefined instantiation of Ada.Containers.Vectors for Integer and this would be probably the least surprising, especially in the context of Y : Unbounded_String, but at the same time I'm open to consider some more lightweight alternatives. The problem is - any other alternative I can think of leads to the complication of API in the form of introducing Ada.Finalization.Controlled to the picture.

Do you have some favourite solutions?

For comparison, in C++ this would be solved with std::vector<int>, which suggests Ada.Containers.Vectors for Ada; but in Java this would be done simply with int[], which is conceptually lighter and easier to use in this particular context.

From: Simon Wright
<simon@pushface.org>
Date: Mon, 08 Oct 2012 12:13:46 +0100
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> [predefined instantiation of Ada.Containers.Vectors]

This would be my choice!

> The problem is - any other alternative I can think of leads to the complication of API in the form of introducing Ada.Finalization.Controlled to the picture.

You could use "private with".

From: Egil Høvik
<egilhovik@hotmail.com>
Date: Mon, 8 Oct 2012 04:29:26 -0700
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> […]

If you don't mind using Ada 2012 packages, there's also Ada.Containers.Indefinite_Holders (RM A.18.18). I haven't used it myself yet, so I can't say anything about how lightweight it is compared to Vector.

From: Maciej Sobczak
<maciej@msobczak.com>
Date: Mon, 8 Oct 2012 04:53:15 -0700
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> [Ada.Containers.Indefinite_Holders]

That would be a perfectly justified choice, except for the fact that this is a library code, which is intended for wider use and relying on Ada 2012 would be too constraining.

Still, a possible approach would be to (re)implement Indefinite_Holders as a supporting package and at some point in the future switch to the standard package with minimal or no impact. I will seriously take it into consideration.

From: Tero Koskinen
<tero.koskinen@iki.fi>
Date: Mon, 8 Oct 2012 17:37:58 +0300
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> […]

I have variation of Indefinite_Holders implemented in Ada 95 here:

https://bitbucket.org/tkoskine/hauki/src/tip/src/hauki-containers-object_holders.ads

https://bitbucket.org/tkoskine/hauki/src/tip/src/hauki-containers-object_holders.adb

My implementation probably doesn't match Ada 2012 100%, but shows that it is possible to implemented them in Ada 95 also.

From: Stephen Leake
<stephen_leake@stephe-leake.org>
Date: Mon, 08 Oct 2012 23:10:48 -0400
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> […]

Ada.Containers.Bounded_Vectors meets these requirements.

From: Shark8
<onewingedshark@gmail.com>
Date: Mon, 8 Oct 2012 19:02:36 -0700
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> […]

Why not use discriminants on the record?

```
Default : constant := Natural'First;
type R (Array_Length : Natural :=
          Natural'First) is record
  X : Integer;
  Y : Unbounded_String;
  Z : Array_Of_Integers_That_We_
    Talk_About_Here (1 .. Array_Length) :=
        (others => Default);
end record;
```

From: Adam Beneschan
<adam@irvine.com>
Date: Mon, 8 Oct 2012 22:16:28 -0700
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> [record discriminant]

Note that this will not work on all implementations. There are two ways to implement an array component whose bound depends on a discriminant: (1) the record type will contain space for the largest possible array, or (2) the record type can contain a pointer to a dynamically sized array, which then may need to be deallocated and reallocated when a new value is assigned to the record. (Other methods are possible in theory.) The above won't work well on compilers that implement (1) because the record will then need to be big enough to hold Natural'Last integers (plus the other fields), and that could be way too much space depending on the range of Natural. If there is a known, fairly small upper bound, and you're willing to allocate that much space in every record, then declare a subtype of Natural and use that subtype in the discriminant clause. Otherwise I'd avoid this construct.

(P.S. Is it just me, or is anyone else irritated by the use of Natural'First or Positive'First? Using something like Integer'First is a great idea, because that makes things portable between implementations since not all implementations have the same Integer bounds. And if you have a subtype S, you should usually say S'First instead of coding the literal value, because who knows what reason you might have in the future for deciding to change the declaration of S, and then you only have to change it in one place. But I don't see the point of Natural'First. It will always be 0, on all implementations, and can never be changed no matter how many times you redesign your program, so why not just write 0? OK, that's my one rant for the day.)

From: Niklas Holsti
<niklas.holsti@tidorum.fi>
Date: Tue, 09 Oct 2012 08:42:08 +0300
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> [record discriminant]

Requirement 4 is not satisfied: if you assign a new value to R.Z, it has to have the same length as the old value (R.Array_Length). You can change the length only by assigning a new value, with the new discriminant value, to the whole record.

And there is the memory usage problem that Adam described.

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 8 Oct 2012 22:45:49 -0700
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> [irritated by Natural'First]

I'm more irritated there's no 'type attribute so you can't say something like Type K(X : Enum_21:= X'Type'First) […] for specifying that you do want the type to have a defaulted discriminant, and that should be the 'First of that type. {After-all the type of X has to be known at compile-time.} Though that particular example might not be the best; using X before it's declaration is illegal. — Though the idea could be useful other places.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Tue, 9 Oct 2012 00:47:12 -0700
Subject: Re: API design considerations - variable-length array in record type
Newsgroups: comp.lang.ada

> Ada.Containers.Bounded_Vectors meets these requirements.

No, because there is no predefined upper bound on the length of array.

I have to admit that the idea with Indefinite_Holders looks the most attractive and the essential parts of this solution are very easy to implement.

How large were MIL STD 1750A Ada-running systems?

From: mjsilva@scriptoriumdesigns.com
Date: Thu, 11 Oct 2012 10:57:09 -0700
Subject: How large were MIL STD 1750A Ada-running systems?
Newsgroups: comp.lang.ada

I wondered about this in another thread, but thought I'd start a separate thread for visibility. How large (how much memory) were 1750A systems that ran Ada (and especially, Ada with some form of tasking)?

I'm only asking about 1750A systems because it was a fairly small device with, from what I read, a lot of Ada activity. What I'm really curious to know is how much memory should be required on a modern 32-bit device to run a small Ada application with e.g. Ravenscar tasking, but without the entire runtime support that full Ada requires.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Thu, 11 Oct 2012 23:03:57 +0300
Subject: Re: How large were MIL STD 1750A Ada-running systems?
Newsgroups: comp.lang.ada

> […]

I worked on one 1750A application, at Space Systems Finland Ltd: the on-board Application SW for the GOMOS ozone-monitoring instrument on the ESA ENVISAT satellite (which was a mostly-or-completely Ada satellite, as I remember).

I think this system had the full 64 kilo-words (128 KiB) of code memory but only 32 kilo-words (64 KiB) of data memory. We used the TLD Ada compiler, but (by the customer's decision) a small RT kernel instead of Ada tasking.

As I remember, some other 1750A systems on ENVISAT needed more memory, and had to use banking schemes to extend the 16-bit address space (with the predictable problems that causes).

> [Ravenscar tasking]

Sorry, I have no experience of Ada tasking on the 1750A. But it seems to me that the RTOS we used was about as complex as Ravenscar tasking, and it was definitely not the major consumer of code memory.

RESTful web API using AWS

From: Adrian Hoe <abyhoe@gmail.com>
Date: Thu, 11 Oct 2012 08:17:59 -0700
Subject: Re: RESTful web API using AWS?
Newsgroups: comp.lang.ada

> I'm currently tasked with developing a RESTful web API as a font-end for what amounts to a bunch of bog-standard RDB operations. I'm currently using PHP, which, as it stands is making me feel ill.

> While investigating other (more orthogonal) technology choices

> (erlang, Scala, etc.), I happened to come across Ada Web Server. I suppose my question is: is AWS up to the job? Has anyone here done this before?

> It seems to me like AWS' dispatcher mechanisms are ideal for this purpose, because they don't make assumptions about what you want to do with the incoming HTTP request (I'm tired of frameworks that force you into a ham-fisted, code generating MVC implementation).

> Any input from experienced Ada engineers would be greatly appreciated.

Yes, I'm developing both web application servers and RESTful web API servers using AWS. Codes written in Ada are more maintainable than anything else especially when your apps are big.

I strongly recommend. Cheers.

High-precision floating point

From: Jerry <lanceboyle@qwest.net>
Date: Fri, 19 Oct 2012 01:57:19 -0700
Subject: How to get high-precision floating point--MPFR
Newsgroups: comp.lang.ada

I have a few lines of floating point code in a large program which I suspect could be causing subtle errors. (It uses lots of recursion with some dicey numbers.) I am using GNAT GPL 2011 on an Intel Mac, Long_Float (64 bits) everywhere.

I understand that recent versions of GCC come with MPFR (Multiple Precision Floating-Point Reliably).

Here is what I want to do

```
type Giant_Float is digits 30;
```

but GNAT says, sorry not more than digits 18 (see below related note). So what I think I have to do is find an Ada binding to MPFR and re-write my code. Is that correct? Do I already have the library with GPL 2011 or OS X 10.7? If not, I can build with Macports. I gather that Vincent's binding here

http://code.google.com/p/adabindinggmpmpfr/

is the way to go. Did this binding ever make it into GCC?

If I do quadruple precision I would expect the relevant lines to execute 5-6 times slower than with Long_Float. I need only multiplication and addition, and will convert the result to Long_Float.

I don't need a computational model where the precision increases every time I do an operation; I'm certain that I can easily specify a fixed precision that will suffice. Is that the way MPFR works?

On a related note, I have a (borrowed/stolen) program which prints out some numeric attributes. I won't bother pasting the code, but a partial output listing is this:

```
Long_Float bits is 64

Long_Long_Float bits is 128
```

The smallest Long_Float is 1.94469227433160678E-62

The largest Long_Float is 2.57110087081438330E+61

The number of digits in Long_Float is 15

The size of the Long_Float mantissa in bits is 51

However, the CPU's Long_Float mantissa is 53

The smallest Long_Long_Float is 1.76868732008334226E-74

The largest Long_Long_Float is 2.82695530364541493E+73

The number of digits in Long_Long_Float is 18

The size of the Long_Long_Float mantissa in bits is 61

However, the CPU's Long_Long_Float mantissa is 64

Can someone explain this? At first it looks like Long_Long_Float is going to be quadruple precision (128 bits) but then the precision results indicate that it is only 3 digits better than Long_Float and has a rather small increase in exponent range. And why is the Ada type using less mantissa bits than the hardware?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 19 Oct 2012 11:45:41 +0200
Subject: Re: How to get high-precision floating point--MPFR
Newsgroups: comp.lang.ada

AFAIK x86/x86-64 do not have hardware 128-bit floats (IEEE or not). See

http://en.wikipedia.org/wiki/Quadruple-precision_floating-point_format

Which also says that gcc's long double is merely 80-bit extended precision. That, I presume, corresponds to the GNAT's Long_Long_Float.

You could compare attributes of Long_Long_Float and Interfaces.C.long_double, T'Machine_Mantissa (A.5.3), in particular, it should be 64 for 80-bit extended precision.

Why GNAT uses 128 bit instead of 80 actually used bit? Maybe, because of alignment issues.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Fri, 19 Oct 2012 12:47:53 +0200
Subject: Re: How to get high-precision floating point--MPFR
Newsgroups: comp.lang.ada

One possible solution has a description here:

http://web.am.qub.ac.uk/users/j.parker/miscellany/arbitrary/README.arbitrary

From: Jerry <lanceboyle@qwest.net>
Date: Fri, 19 Oct 2012 04:05:28 -0700
Subject: Re: How to get high-precision floating point--MPFR
Newsgroups: comp.lang.ada

That looks interesting. The readme says it is done entirely in integers but suitably optimized runs 1/2 to 2/3 the speed of Fortran quad precision.

Ada.Storage_IO example?

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Tue, 23 Oct 2012 21:42:43 +0200
Subject: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

I'm seeking for a tiny use case showing `Ada.Storage_IO` in action. I also believe the topic is worth for many people, beginners or not, as this is the kind of stuff so typical of Ada. I always liked the clean distinction Ada makes between streams and files. Looking for some reminder about it in the RM, I just noticed this `Ada.Storage_IO` I've never noticed before, but feel surprised the buffer always contains a single element; thus the question. The RM makes a reference from there to `Ada.Direct_IO`, but the latter does not reciprocally mention the former.

From: Jeffrey Carter <jrcarter@acm.org>
Date: Tue, 23 Oct 2012 14:02:43 -0700
Subject: Re: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

> […]

The buffer is the correct size that corresponds to an object of Element_Type. Much easier than figuring that out oneself.

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 23 Oct 2012 14:21:21 -0700
Subject: Re: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

> […]

Looking at the AARM, it appears that the main reason this might be useful is you've defined a record type that contains a hidden pointer (i.e. a pointer that the Ada implementation sets up, not a subcomponent of an access type). In that case, the buffer contains the "flattened" version of the data. E.g.

```
type Rec is record
  S : String (1 .. Name_Size);
  … other components
end record;
```

where Name_Size is non-static object (i.e. the value can't be determined at compile time). Some implementations (but not GNAT, I believe) would store S as a pointer, rather than fool around with Rec objects whose size isn't known at compile time, and possibly components whose location in the record isn't known at compile time. In this case, you couldn't rely on Rec'Size (or Rec'Size / System.Storage_Unit) to be large enough to contain all the data; I think that's the kind of case where the language designers thought this would be useful, because you could write code that would work with any Ada implementation. Still, you should be able to accomplish the same thing using the stream features of Ada, and I think that would be preferable.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 24 Oct 2012 07:03:43 +0200
Subject: Re: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

> […]

Imagine you want to implement a binary IO package. On one side you have high level data type, on the other side system files that are just blocks of bytes. Storage_IO allows you to bridge this gap (at least that's my understanding).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 24 Oct 2012 09:34:58 +0200
Subject: Re: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

> […]

In that case the package should rather define conversions to and from Storage_Array, better as in-place operations with a position to get/put data at. Furthermore the generic parameter should have been declared as:

```
type Element_Type (<>) is private.
```

And, of course, implementation-defined I/O is a bad idea anyway in modern times of interoperability. (This also applies to Direct_IO and Sequential_IO, which are remnants of late 70's.)

From: Christoph Grein <christ-usch.grein@t-online.de>
Date: Wed, 24 Oct 2012 02:49:02 -0700
Subject: Re: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

See AARM A.9(1.a):

Reason: This package exists to allow the portable construction of user-defined direct-access-oriented input-output packages. The Write procedure writes a value of type Element_Type into a Storage_Array of size Buffer_Size, flattening out any implicit levels of indirection used in the representation of the type. The Read procedure reads a value of type Element_Type from the buffer, reconstructing any implicit levels of indirection used in the representation of the type. It also properly initializes any type tags that appear within the value, presuming that the buffer was written by a different program and that tag values for the "same" type might vary from one executable to another.

> Furthermore the generic parameter should have been declared as:

> type Element_Type (<>) is private.

See A.9(10.a):

Reason: As with Direct_IO, the Element_Type formal of Storage_IO does not have an unknown_discriminant_part so that there is a well-defined upper bound on the size of the buffer needed to hold the content of an object of the formal subtype (i.e. Buffer_Size). If there are no implicit levels of indirection, Buffer_Size will typically equal:

```
(Element_Type'Size +
         System.Storage_Unit - 1) /
         System.Storage_Unit
```

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 24 Oct 2012 12:28:00 +0200
Subject: Re: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

portable construction of X /= construction of portable X

The package cannot be used to write files in a way that an application compiled by one compiler X1 under OS Y1 could write a file readable for compiler X2 under OS Y2.

In order to be portable they should have defined the exact representation of the object in the file. Yes, nobody would do that, but then they should not include useless packages into the standard either.

> […]

The reason why Direct_IO must have an upper bound is to allow accessing records randomly without an excessive (in 70's it was excessive) overhead of indexing them. This simply does not apply to Storage_IO, which has just one "record."

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 24 Oct 2012 14:27:28 +0200
Subject: Re: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

> What about private persistent storage?

X1 = X2 and Y1 = Y2

What are chances of having this? And what are chances that a user would mistakenly use the package when X1 /= X2 or Y1 /= Y2?

Specifically to persistent storage, you would need to be able to debug it, reinterpret using some independent tools, have redundancy and additional checks on top, be able to store indefinite objects, be independent on the medium etc. A *generic- package parameterized by a private type with no representation defined is a non-starter.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Wed, 24 Oct 2012 16:03:51 +0200
Subject: Re: Ada.Storage_IO: applied example?
Newsgroups: comp.lang.ada

> What about private persistent storage?

If traditional Ada I/O is good for your Ada program, you can use Storage_IO in conventional algorithms this way. The setup can employ static polymorphism that allows selecting instances of Storage_IO or Direct_IO as needed. Either at startup or later.

<rant>

The solution, using the egregious generic mechanisms of Ada, will suffer from all the advantages of lack of an unused tagged parameter.

The whole approach of stating requirements as generic formals suffers from good opportunities for compilers to perform optimization.

It suffers from the fact that stored internal program data is not being stored in an ISO-approved form ready for inspection by non-generic programs of an unrelated make and purpose.

It also suffers from the problem that it still cannot make the external world an Ada object, which could mollify the previously argued flaw.

It suffers from lack of opportunities to control I/O of octets algorithmically, thus preventing emulation of nice per-program Storage_IO objects in DIY fashion.

It also suffers from not being a silver bullet.

But other than that …</rant>

```
with Ada.Direct_IO;
with Ada.Storage_IO;
with Ada.Command_Line;
use Ada.Command_Line;
procedure Lookahead is
  use Ada;
  type Sentence (Length : Positive := 100)
    is record
    Text : Wide_String (1 .. 100);
  end record;
  subtype Name is Wide_String (1 .. 20);
  type Big is record
    -- objects that will be processed
    First, Last : Name;
    Age : Natural;
    Motto : Sentence;
  end record;
  generic
    with procedure Store_Single_Item
           (Item : in Big);
    with procedure Fetch_Single_Item
           (Item : out Big);
  procedure Process_Records;
  procedure Process_Records is separate;

begin
  if Natural'Value (Argument(1)) = 0 then
    declare
      package Volatile is
        procedure Store (Item : in Big);
        procedure Fetch (Item : out Big);
      end Volatile;
      package body Volatile is
        package Undo_Buffer is new
           Storage_IO (Big);
        Storage : Undo_Buffer.Buffer_Type;
        procedure Fetch (Item : out Big) is
        begin
          Undo_Buffer.Read (Storage, Item);
        end Fetch;
        procedure Store (Item : in Big) is
        begin
          Undo_Buffer.Write (Storage, Item);
        end Store;
      end Volatile;
      procedure Run is new
        Process_Records
        (Store_Single_Item => Volatile.Store,
         Fetch_Single_Item => Volatile.Fetch);
    begin
      Run;
    end;
  else
    declare
      package Permanent is
        procedure Store (Item : in Big);
        procedure Fetch (Item : out Big);
      end Permanent;
      package body Permanent is
        package Undo_Buffer is new
           Direct_IO (Big);
        Storage : Undo_Buffer.File_Type;
        procedure Fetch (Item : out Big) is
        begin
          Undo_Buffer.Read (Storage, Item);
        end Fetch;
        procedure Store (Item : in Big) is
        begin
          Undo_Buffer.Write (Storage, Item);
        end Store;
        use Undo_Buffer;
      begin
        open (storage, Inout_File, "item.sto");
      end Permanent;
      procedure Run is new
       Process_Records
       (Store_Single_Item =>
          Permanent.Store,
        Fetch_Single_Item =>
          Permanent.Fetch);
    begin
      Run;
    end;
  end if;
end Lookahead;

separate (Lookahead)
procedure Process_Records is
  package Rec_IO is new Direct_IO (Big);
  use Rec_IO;
  Current: Big := (Age => Natural'Last,
           others => <>);
  input : File_Type;
  output : File_Type;
begin
  open (input, In_File, "records.all");
  open (output, Out_File, "records.flt");
  loop
    Store_Single_Item (Current);
    Read (input, current);
    -- … filter based on pairs …
      write (output, current);
  end loop;
end Process_Records;
```

Import a constant value from C

From: Enzo Guerra <enzoguerra1@gmail.com>
Date: Fri, 26 Oct 2012 15:45:29 -0700
Subject: pragma import (C, ,); -- import a constant value from C library
Newsgroups: comp.lang.ada

Trying to import a constant value from C library (libgsl) but having no luck

```
M_EULER : Long_Float;
-- M_EULER, Euler's constant, \gamma
pragma Import (C, M_EULER,
        "M_EULER");
```

Get error undefined reference to `M_EULER'

From: Simon Wright <simon@pushface.org>
Date: Sat, 27 Oct 2012 00:06:06 +0100

Subject: Re: pragma import (C, ,); -- import a constant value from C library
Newsgroups: comp.lang.ada

> […]

I wonder if the C M_EULER is in fact a macro? If so, it doesn't exist in the library as a symbol in its own right.

You might try defining it as a variable in C and importing that:

```
/- m_euler_definition.c */
#include <libgsl.h>
/- wherever M_EULER's defined */
const double m_euler_for_GNAT =
          M_EULER;
```

-- m_euler.ads

```
M_EULER : constant Long_Float;
-- or Interfaces.C.double
pragma Import (C, "M_EULER",
          "m_euler_for_GNAT");
```

and then when you build you'll need to include m_euler_definition.o.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 27 Oct 2012 09:40:01 +0200
Subject: Re: pragma import (C, ,); -- import a constant value from C library
Newsgroups: comp.lang.ada

[…] in general, you can define any constant this way in Ada. The precision is limited only by the source where you get its tabulated value. And it will work for all real types. E.g.

```
gamma : constant :=
0.57721_56649_01532_86060_65120_9008
2_40243;
X : Long_Float := gamma;
Y : Float := gamma;
…
```

The compiler truncates the value when you use a narrower target like Long_Float.

You can even perform simple calculations with such constants keeping their huge precision. The compiler uses arbitrary-precision arithmetic for them. E.g.

[with Ada.Numerics; use Ada.Numerics;]

```
half_pi : constant := pi / 2.0;
-- far more than enough right digits for a
Long_Float
```

Child packages named Ada

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Wed, 31 Oct 2012 09:47:04 -0700
Subject: Child packages named Ada illegal?
Newsgroups: comp.lang.ada

I was getting misleading <<missing "with Ada…">> errors from GNAT (the packages were withen). In a moment of inspiration I renamed package AA.Languages.Ada to AA.Languages.Ada_Languages and all went well.

Just curious: was GNAT misbehaving (unlikely) or is it somehow forbidden to have child packages named Ada?

From: Shark8 <onewingedshark@gmail.com>
Date: Wed, 31 Oct 2012 10:02:44 -0700
Subject: Re: Child packages named Ada illegal?
Newsgroups: comp.lang.ada

It's probably an issue of visibility. Consider:

```
with Ada.Calendar.Time;
procedure Test is
  package Ada is
  end Ada;

  Now : constant Ada.Calendar.Time :=
        Ada.Calendar.Clock;
begin
  null;
end Test;
```

In this case the Now declaration is prevented from being valid because the type, Time, resides in the package Calendar which is a child of Ada… but Ada at that point refers to Test.Ada which has no Calendar child package.

That the following compiles indicates Ada is not reserved:

```
procedure Test is
  Now : constant Ada.Calendar.Time :=
        Ada.Calendar.Clock;
begin
  declare
    package Ada is
    end Ada;
  begin
    null;
  end;
end Test;
```

From: Adam Beneschan <adam@irvine.com>
Date: Wed, 31 Oct 2012 10:20:31 -0700
Subject: Re: Child packages named Ada illegal?
Newsgroups: comp.lang.ada

Without seeing your exact code, it's hard to say for sure, but Shark8 is probably right that it's a visibility issue. If you're inside AA.Languages.Ada, and you refer to the identifier Ada without any other qualification, then that identifier would refer to your child package, not the top level Ada package. (Standard.Ada would still get you the top-level package.) The error messages are confusing, though. I can understand it, because 99.9999% of the time when a user uses one of the language-defined package names and it isn't defined, it's because they forgot to WITH it; the GNAT people apparently didn't think of this possibility, which is understandable.

From: Shark8 <onewingedshark@gmail.com>
Date: Wed, 31 Oct 2012 12:39:32 -0700
Subject: Re: Child packages named Ada illegal?
Newsgroups: comp.lang.ada

> package body AA.Languages.Ada is
> …
> function
>Non_Alphanum_To_Underscore
> (From >: Character) return Character is
> (if Ada.Characters.Handling.
> Is_Alphanumeric (From) then From else '_');
> … end;

One option would be to put package renaming inside the AA or Languages package; thus:

```
with Ada.Characters;
package AA.Language is
…
package Internal_Characters
          renames Ada.Characters;
end AA.Language;
```

would allow you to refer to Ada.Characters via the Internal_Characters name.

Class wide preconditions

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Mon, 05 Nov 2012 21:41:48 +0100
Subject: Class wide preconditions: error in the Ada 2012 Rationale?
Newsgroups: comp.lang.ada

I have a doubt, and this one disturb me, so this topic.

I was reading the revised version (fourth draft) of the Ada 2012 Rationale, when I saw this:

> However, the rules regarding preconditions are perhaps surprising. The specific precondition Pre for Equilateral_Triangle must be true (checked in the body) but so long as just one of the class wide preconditions Pre'Class for Object and Triangle is true then all is well.

Then later, a summary of the rule:

> In summary, class wide preconditions are checked at the point of call. Class wide postconditions and both specific pre- and postconditions are checked in the actual body.

http://www.ada-auth.org/standards/12rat/html/Rat12-2-3.html

I believe either my understanding is wrong, or the Rationale is wrong. The above statements are not compatible with the substitution principle. What if a subprogram expects a class wide type with a root type and its precondition, and get a derived type with a specific precondition it can't know about?

There may be a comment on that point, see below, but first, an example test, compiled with GNAT:

```
with Ada.Text_IO;
procedure Ex is
  package P1 is
```

```
    type T is interface;
    Condition : Boolean;
    procedure P (E : T) is abstract
          with Pre'Class => P1.Condition;
  end P1;

  package P2 is
    type T is interface and P1.T;
    Condition : Boolean;
    overriding
    procedure P (E : T) is abstract
          with Pre'Class => P2.Condition;
  end P2;

  package P3 is
    type T is new P2.T with null record;
    Condition : Boolean;
    overriding
    procedure P (E : T)
    with Pre'Class => P3.Condition;
  end P3;

  package body P3 is
    procedure P (E : T) is
    begin
      Ada.Text_IO.Put_Line ("You're OK!");
    end P;
  end P3;

  E : P3.T;

begin
  P1.Condition := True;
  P2.Condition := True;
  P3.Condition := True;
  -- Hint: try to set this to `False`.
  E.P;
end Ex;
```

Side-note: the package prefix added to access the condition in the Pre'Class, are not strictly required, but required with GNAT to bypass one of its bug.

Test this example modifying each of the conditions: it `P3.Condition` is set to `False`, there won't be any error at runtime, and you'll be OK, although `P3.Condition` is the specific precondition for the concrete implementation. It will fails only if all preconditions are false. No specific precondition seems to be checked in the body of `P`, or else it would trigger an exception when `P3.Condition` is set to `False`.

GNAT's interpretation is compatible with my own, and does not seems to be compatible with the one from the Rationale.

But there may be a trick? Let's try turning `E.P;` into `P3.P (E);`: same result. Not let's try turning `P3.P (E);` into `P3.P (P3.T'(E));`: same result. GNAT does not know about anything like specific precondition, and I'm OK with GNAT this time.

So is this me not understanding the Rationale? Or the Rationale has either wrong or unclear words?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 8 Nov 2012 18:57:51 -0600
Subject: Re: Class wide preconditions: error in the Ada 2012 Rationale?
Newsgroups: comp.lang.ada

>> In summary, class wide preconditions are checked at the point of call. Class wide post-conditions and both specific pre- and post-conditions are checked in the actual body.

The above is correct.

> The above statements are not compatible with the substitution principle.

*Specific- preconditions and postconditions are not necessarily compatible with the substitution principle. If you want that, you either have to be careful what you write, or (better IMHO) use only class-wide preconditions and postconditions.

You don't always want strict LSP, and using specific preconditions gives you a way to get that when needed. But of course, in that case, dispatching calls may fail for no reason visible at the point of the call. (LSP = Liskov Substitutability Principle).

> What if a sub-program expects a class wide type with a root type and its precondition, and get a derived type with a specific precondition it can't know about?

You still evaluate the specific precondition associated with the subprogram that is actually called.

My understanding is that a lot of GNAT users only use carefully written specific preconditions (probably because they learned how to do that before class-wide preconditions existed in GNAT). Those can be, but don't have to, follow LSP. OTOH, class-wide preconditions follow LSP by design.

My rule of thumb is that in a given derivation chain, you should only use one or the other. (I wanted to make that a requirement, but that was shot down.)

I think given the sorts of programs that you write, you should only use class-wide preconditions and postconditions, and forget that specific ones exist at all. In which case, you won't have a problem with LSP.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 8 Nov 2012 20:13:23 -0600
Subject: Re: Class wide preconditions: error in the Ada 2012 Rationale?
Newsgroups: comp.lang.ada

> How shot down? (if the question is not too much inquisitive)

I don't recall the details, but I think most people wanted as few restrictions as possible. "Methodological restrictions" have a bad history in Ada, and thus we often avoid them. Clearly a tool like AdaControl can enforce style rules beyond those the language requires, and that was thought to be the best option here.

IBM 437 encoded String to UTF-16 Wide_String

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue, 27 Nov 2012 13:02:05 -0800
Subject: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

I'm looking for a IBM 437 encoded String [1] to UTF-16 Wide_String conversion.

[1] http://en.wikipedia.org/wiki/Code_page_437

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue, 27 Nov 2012 14:12:04 -0800 (P
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

> […]

Found!

[…]

http://www.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/PC/CP437.TXT

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 27 Nov 2012 23:14:22 +0100
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

> […]

What about

```
type Map is array (Character) of
   Wide_Character :=
         (Wide_Character'Val (0),
          Wide_Character'Val (1),
      <other values from the wiki page> );
```

It is not a big deal to type 256 values. Half of them (0..127) are literals corresponding to 7-bit ASCII.

That would give you UCS-2. I presume that UTF-16 is not needed in this case.

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue, 27 Nov 2012 15:13:00 -0800
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

> It is not a big deal to type 256 values.

Type ? I'm too lazy for that :-).

I've put this:

http://www.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/PC/CP437.TXT

(from the Wiki page) into Excel, and abracadabra, it became that:

http://sf.net/p/azip/code/69/tree//
trunk/gui_common/azip_common.adb

From: Vadim Godunko <vgodunko@gmail.com>
Date: Tue, 27 Nov 2012 15:41:35 -0800
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

Matreshka includes text codecs to convert text data between different encoding, see

http://forge.ada-ru.org/matreshka/wiki/
League/TextCodec

From: Emmanuel Briot <briot.emmanuel@gmail.com>
Date: Wed, 28 Nov 2012 00:34:53 -0800
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

XML/Ada also has a few conversion packages, but not IBM 437.

I think the most convenient here would be to create a small binding to the iconv library. I believe it exists on most systems, although with slightly different interfaces. And it supports a huge number of encodings.

You basically need to bind three functions ("open", "iconv" and "close")

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 28 Nov 2012 09:52:33 +0100
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

> [binding to the iconv library]

No. IMO the most convenient way would be to fix the language in order to have Wide_Wide_String'Class of which String, Wide_String, Wide_Wide_String, UTF8_String etc were members.

Encoding is nothing but an instance of Wide_Wide_String'Class implementing the interface of an array of code units. In the case of IBM 437 it is something like:

```
type IBM_437_String is
   new Wide_Wide_String
    -- Logical view, string of code points
   and array (Positive range <>) of Byte;
   -- Presentation view
```

Conversions if ever needed, would be type/view conversions.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 28 Nov 2012 10:58:59 +0100
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

> If there isn't anything special about {Wide_}Character, a Vector of Character might be an alternative,

No. When I mentioned Wide_Wide_String I meant an array of code points. The logical view of *any-string type is array of code points. The only difference between different string types is in the constraints put on the code points. E.g. String has code points 0 to 255. IBM_437_String would have a non-contiguous set of code points etc.

> though hated by haters of generics, I should think.

I don't see how generics are relevant here. All strings are arrays of code points. They all belong to this class. No explicit conversions should be needed between them.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Wed, 28 Nov 2012 12:31:35 +0100
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

In case of differentiation by sets of code points, I'd rather have an honest type Unicode_String and- if we are already fixing the language- put everything that has {Wide_}String in its name in Annex J.

But then, consider

```
type Index is range 1 .. 12;
type R is ('I', 'V', 'X', 'L', 'C', 'D', 'M');
type N is array (Index range <>) of R;
```

A string of R, named N here, is just fine. In fact,

```
Year : constant N := "MCMLXXXIII";
```

has a valid literal, and the year so written is not of any of the standard string types. The definition of type R actually implies a codespace, and, for example, Character'('V') or Wide_Character'('V') have no role in it, irrespective of any accidental overlap in encoding or representation or position.

So, which by force should type N be in Whatever_String'Class?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 28 Nov 2012 14:36:02 +0100
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

>> IBM_437_String would have a non-contiguous set of code points etc.

> What is a non-contiguous set?

A convex set in this case, i.e.:

for code points x,y,z, such that x<y<z if x,z in S then y in S

> […] So, which by force should type N be in Whatever_String'Class?

Per inheritance:

```
type N is
   new Wide_Wide_String
   and array (...) of R;
```

[…] The problem is not construction of a container type. It is the relation of the obtained type to the string interface. The string interface is an array of code points. The container must implement this interface in order to be a string. All strings must implement this interface, this is why they are called "strings."

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 28 Nov 2012 19:00:46 +0100
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

[…] A string can have any implementation. The set of operations is determined by the contract, not otherwise.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 28 Nov 2012 21:18:23 -0600
Subject: Re: IBM 437 encoded String to UTF-16 Wide_String
Newsgroups: comp.lang.ada

> In case of differentiation by sets of code points, I'd rather have an honest type Unicode_String and --- if we are already fixing the language --- put everything that has {Wide_}String in its name in Annex J.

That's rather what I would like to do, especially as trying to support Wide_Wide_String file names makes things into a hash. (Do we really want to have Wide_Wide_Open in Text_IO??).

The language already has almost everything needed to support Root_String'Class. Most of the missing capabilities center around getting string literals for such a type. We'd probably need to keep Wide_Wide_String around in order to provide a common interconversion format.

> So, which by force should type N be in Whatever_String'Class?

N is not derived from Root_String'Class, and as such it couldn't be used with Put_Line (for one example). If you derived it from that type (possibly using a generic to fill in the operations), then of course you could. In that case, you'd have to provide (or let the generic provide) conversions to and from Unicode.

Conference Calendar

Dirk Craeynest

KU 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.

2013

☺ January 23-25	40th ACM SIGPLAN-SIGACT **Symposium on Principles of Programming Languages** (POPL'2013), Rome, Italy. Topics include: fundamental principles and important innovations in the design, definition, analysis, transformation, implementation and verification of programming languages, programming systems, and programming abstractions.
	Jan 20-21 ACM SIGPLAN **Workshop on Partial Evaluation and Program Manipulation** (PEPM'2013). Topics include: Program and model manipulation techniques (such as: partial evaluation, slicing, symbolic execution, refactoring, ...); Program analysis techniques that are used to drive program/model manipulation (such as: abstract interpretation, termination checking, type systems, ...); Techniques that treat programs/models as data objects (including: metaprogramming, generative programming, model-driven program generation and transformation, ...); etc. Application of the above techniques including case studies of program manipulation in real-world (industrial, open-source) projects and software development processes, descriptions of robust tools capable of effectively handling realistic applications, benchmarking.
Jan 29 - Feb 01	11th **Australasian Symposium on Parallel and Distributed Computing** (AusPDC'2013), South Australia. Topics include: multicore systems; GPUs and other forms of special purpose processors; middleware and tools; parallel programming models, languages and compilers; runtime systems; reliability, security, privacy and dependability; applications; etc.
♦ February 03	**Ada at the Free and Open-Source Software Developers' European Meeting** (FOSDEM'2013), Brussels, Belgium. FOSDEM 2013 is a two-day event (Sat-Sun 02-03 February). This years' edition includes again an Ada Developer Room, organized by Ada-Belgium in cooperation with Ada-Europe, which will be held on Sunday 3 February.
January 21-23	6th **India Software Engineering Conference** (ISEC'2013), New Delhi, India. Topics include: static analysis, specification and verification, model driven software engineering, software architecture and design, tools and environments, maintenance and evolution, component based software engineering, object-oriented technology, distributed software development, software engineering education, software security, mining software repositories, embedded and real-time systems, etc.
☺ January 23-27	18th ACM SIGPLAN **Symposium on Principles and Practice of Parallel Programming** (PPoPP'2013), Shenzhen, China. Topics include: formal analysis and verification; parallel programming languages; compilers and runtime systems; development, analysis, or management tools; concurrent data structures; synchronization and concurrency control; software engineering for parallel programs; software issues for multicore and multithreaded processors; task mapping and scheduling; etc.
Feb 27 – Mar 01	5th **International Symposium on Engineering Secure Software and Systems** (ESSoS'2013), Paris, France. Topics include: security architecture and design for software and systems; specification formalisms for security artifacts; verification techniques for security properties; systematic support for security best practices; programming paradigms for security; processes for the development of secure software and systems; support for assurance, certification and accreditation; etc.

Feb 27 – Mar 01 21st Euromicro **International Conference on Parallel, Distributed and Network-Based Computing** (PDP'2013), Belfast, Northern Ireland, UK. Topics include: embedded parallel and distributed systems, multi- and many-core systems, programming languages and environments, runtime support systems, dependability and survivability, advanced algorithms and applications, etc.

March 05-08 17th **European Conference on Software Maintenance and Reengineering** (CSMR'2013), Genova, Italy. Topics include: development of maintainable systems, and evolution, migration and reengineering of existing ones; experience reports on maintenance and reengineering of large-scale systems; language support for software maintenance and evolution; etc.

March 06-09 44th ACM **Technical Symposium on Computer Science Education** (SIGCSE'2013), Denver, Colorado, USA.

March 16-24 ETAPS2013 - 16th **International Conference on Fundamental Approaches to Software Engineering** (FASE'2013), Rome, Italy. Topics include: software engineering as an engineering discipline; specification, design, and implementation of particular classes of systems (collaborative, embedded, distributed, ...); software quality (validation and verification of software using theorem proving, model checking, testing, analysis, refinement methods, metrics, ...); model-driven development and model transformation; software evolution (refactoring, reverse and re-engineering, configuration management, ...); etc.

March 18-22 28th ACM **Symposium on Applied Computing** (SAC'2013), Coimbra, Portugal.

☺ Mar 18-22 **Track on Programming Languages** (PL'2013). Topics include: Compiling Techniques, Formal Semantics and Syntax, Garbage Collection, Language Design and Implementation, Languages for Modeling, Model-Driven Development New Programming Language Ideas and Concepts, Practical Experiences with Programming Languages, Program Analysis and Verification, Programming Languages from All Paradigms, etc.

☺ Mar 18-22 **Track on Object-Oriented Programming Languages and Systems** (OOPS'2013). Topics include: Aspects and components; Distribution and concurrency; Formal verification; Integration with other paradigms; Interoperability, versioning and software evolution and adaptation; Language design and implementation; Modular and generic programming; Static analysis; Type systems; etc.

March 18-22 **Track on Software Verification and Testing** (SVT'2013). Topics include: tools and techniques for verification of large scale software systems, real world applications and case studies applying software verification, static and run-time analysis, refinement and correct by construction development, software certification and proof carrying code, etc.

March 18-22 **Embedded Systems Track** (EMBS'2013). Topics include: Multithreading in Embedded Systems design and development; Compilation strategies, code transformation and parallelization for Embedded Systems; Reliability in Embedded Computing and Systems; Security within Embedded Systems; Safety-critical Embedded Systems; Case studies; etc.

March 18-22 6th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2013), Luxembourg. Topics include: domain specific testing, security testing, embedded-software testing, testing concurrent software, testing large-scale distributed systems, testing in multi-core environments, quality assurance, empirical studies, agile/iterative/incremental testing processes, testing of open source and third-party software, software reliability, formal verification, experience reports, etc. Deadline for submissions: mid January 2013 (workshop contributions), February 9, 2013 (posters).

March 18 2nd **International Workshop on Engineering Safety and Security Systems** (ESSS'2013). Topics include: methods, techniques and tools for system safety and security; methods, techniques and tools for analysis, certification, and debugging of complex safety and security systems; case studies and experience reports on the use of formal methods for analyzing safety and security systems; etc.

March 23 ETAPS2013 - 10th **International Workshop on Formal Engineering approaches to Software Components and Architectures** (FESCA'2013), Rome, Italy. Topics include: modelling formalisms for the analysis of concurrent, embedded or model-driven systems assembled of components; interface compliance (interface-to-interface and interface-to implementation) and contractual use of components;

techniques for prediction and formal verification of system properties, including static and dynamic analysis; industrial case studies and experience reports; etc.

March 25-29 — 12th **International Conference on Aspect-Oriented Software Development** (AOSD'2013), Fukuoka, Japan. Topics include: Complex systems; Software design and engineering (evolution, economics, composition, methodology, ...); Programming languages (language design, compilation and interpretation, verification and static program analysis, formal languages, execution environments and dynamic weaving, ...); Varieties of modularity (model-driven development, generative programming, software product lines, contracts and components, ...); Tools (evolution and reverse engineering, crosscutting views, refactoring, ...); Applications (distributed and concurrent systems, middleware, runtime verification, ...); etc.

April 01-05 — 6th **Latin-American Symposium on Dependable Computing** (LADC'2013), Rio de Janeiro, Brazil. Topics include: computer system dependability. Deadline for submissions: January 18, 2013 (fast abstracts, student forum and industrial track).

April 08-11 — 25th IEEE **Software Technology Conference** (STC'2013), Salt Lake City, Utah, USA. Theme: "Back to the Future: Enabling Mission Success through Software Technology". Topics include: cybersecurity, software production efficiencies, resilient software, software engineering best practices, etc.

♦ April 17-19 — 16th **International Real-Time Ada Workshop** (IRTAW'2013), Kings Manor, York, UK. In cooperation with Ada-Europe. Deadline for submissions: February 1, 2013 (position papers).

☺ May 18-26 — 35th **International Conference on Software Engineering** (ICSE'2013), San Francisco, USA. Theme: "Software Engineering Ideas to Change the World". Deadline for submissions: January 30, 2013 (SCORE full project submission).

☺ May 20-24 — 27th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2013), Boston-Cambridge, 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, parallel programming paradigms, and programming environments and tools, etc.

June 04-06 — 13th **International Conference on Software Process Improvement and Capability dEtermination** (SPICE'2013), Bremen, Germany. Topics include: process assessment, improvement and risk determination in areas of application such as automotive systems and software, aerospace systems and software, medical device systems and software, safety-related systems and software, financial institutions and banks, small and very small enterprises, etc.

June 04-07 — 22nd **Australasian Software Engineering Conference** (ASWEC'2013), Melbourne, Australia. Topics include: empirical research in software engineering; formal methods; legacy systems and software maintenance; measurement, metrics, experimentation; modularisation techniques, including component-based software engineering and aspect-oriented programming; programming techniques, such as object-oriented, functional and hybrid programming; open source software development; quality assurance; real-time and embedded software; software analysis; software design and patterns; software engineering education; software processes and quality; software re-use and product development; software risk management; software security, safety and reliability; software verification and validation; standards and legal issues; etc. Deadline for submissions: March 1, 2013 (experience reports, doctoral symposium).

♦ June 10-14 — 18th **International Conference on Reliable Software Technologies - Ada-Europe'2013**, Berlin, Germany. Sponsored by Ada-Europe, in cooperation with ACM SIGAda, SIGBED, SIGPLAN. Deadline for submissions: January 14, 2013 (industrial presentations).

June 10-14 — 10th **International Conference on integrated Formal Methods** (iFM'2013), Turku, Finland. Topics include: the combination of (formal and semi-formal) methods for system development, regarding modeling and analysis, and covering all aspects from language design through verification and analysis techniques to tools and their integration into software engineering practice. Deadline for submissions: January 10, 2013 (abstracts), January 17, 2013 (papers).

June12-14	14th **International Conference on Product Focused Software Development and Process Improvement** (PROFES'2013), Paphos, Cyprus. Topics include: software engineering techniques, methods, and technologies for product-focused software development and process improvement as well as their practical application in an industrial setting. Deadline for submissions: January 31, 2013 (papers).
June 16-21	ACM SIGPLAN **Conference on Programming Language Design and Implementation** (PLDI'2013), Seattle, Washington, USA. Topics include: programming languages, their design, implementation, development, and use; innovative and creative approaches to compile-time and runtime technology, novel language designs and features, and results from implementations; language designs and extensions; static and dynamic analysis of programs; domain-specific languages and tools; type systems and program logics; checking or improving the security or correctness of programs; memory management; parallelism, both implicit and explicit; debugging techniques and tools; etc.
June 18-21	13th **International Conference on Software Reuse** (ICSR'2013), Pisa, Italy. Theme: "Safe and Secure Reuse". Topics include: guaranteeing safety and security related properties of reusable components, certification issues for mission-critical reusable components, component-based reuse, COTS-based development, generator-based techniques, domain-specific languages, testing in the context of software reuse, model-driven development, reuse of non-code artifacts (process, experience, etc.), software product line techniques, quality-aspects of reuse, industrial experience with reuse, software evolution and reuse, large-scale systems, etc.
☺ June 19-21	16th IEEE **International Symposium on Object/component/service-oriented Real-time distributed Computing** (ISORC'2013), Paderborn, Germany. Topics include: Programming and system engineering (ORC paradigms, languages, model-driven development of high integrity applications, specification, design, verification, validation, testing, maintenance, ...); System software (real-time kernels, middleware support for ORC, extensibility, synchronization, scheduling, fault tolerance, security, ...); Applications (embedded systems (automotive, avionics, consumer electronics, ...), real-time object-oriented simulations, ...); System evaluation (timeliness, worst-case execution time, dependability, end-to-end QoS, fault detection and recovery time. ...); etc.
July 01-03	18th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2013), Canterbury, Kent, UK.
July 01-03	7th **International Symposium on Theoretical Aspects of Software Engineering** (TASE'2013), Birmingham, UK. Topics include: the theoretical aspects of model driven software engineering, component based software engineering, software security, reliability, and verification, embedded and real time software systems, aspect and object oriented software design, reverse engineering, etc. Deadline for submissions: January 18, 2013 (abstracts), January 25, 2013 (papers).
July 01-05	27th **European Conference on Object-Oriented Programming** (ECOOP'2013), Montpellier, France. Topics include: all areas of object technology and related software development technologies, such as aspects, components, modularity, concurrent and parallel systems, distributed computing, programming environments, versioning, refactoring, software evolution, language definition and design, language implementation, compiler construction, design methods and design patterns, real-time systems, security, specification, verification, type systems, etc. Deadline for early registration: June 1, 2013.
July 03-07	8th **International Conference on Evaluation of Novel Approaches to Software Engineering** (ENASE'2013), Angers, France. Topics include: emerging as well as established SE methods, practices, architectures, technologies and tools; software process improvement, model-driven engineering, application integration technologies, software quality management, software change and configuration management, geographically distributed software development environments, formal methods, component-based software engineering and commercial-off-the-shelf (COTS) systems, software and systems development methodologies, etc. Deadline for submissions: January 30, 2013 (papers), March 31, 2013 (mini-symposia).
July 17-19	18th IEEE **International Conference on the Engineering of Complex Computer Systems** (ICECCS'2013), Singapore. Topics include: verification and validation, security of complex systems, 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 submissions: February 1, 2013 (abstracts), February 15, 2013 (papers, workshops).

July 23-25	25th ACM **Symposium on Parallelism in Algorithms and Architectures** (SPAA'2013), Montreal, Canada. Topics include: parallel and distributed algorithms; multi-core architectures; compilers and tools for concurrent programming; synergy of parallelism in algorithms, programming, and architecture; etc. Deadline for submissions: February 11, 2013 (abstracts), February 13, 2013 (full papers).
September 08-11	10th **International Conference on Parallel Processing and Applied Mathematics** (PPAM'2013), Warsaw, Poland. Topics include: multi-core and many-core parallel computing; parallel/distributed algorithms: numerical and non-numerical; scheduling, mapping, load balancing; parallel/distributed programming; tools and environments for parallel/distributed computing; security and dependability in parallel/distributed environments; applications of parallel/distributed computing; etc. Event also includes: workshop on Language-Based Parallel Programming Models. Deadline for submissions: February 1, 2013 (workshops), April21, 2013 (papers).
☺ September 10-13	**International Conference on Parallel Computing** 2013 (ParCo'2013), München, Germany. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments, in particular Parallel programming languages, compilers, and environments; Tools and techniques for generating reliable and efficient parallel code; Best practices of parallel computing on multicore, manycore, and stream processors; etc. Deadline for submissions: February 28, 2013 (extended abstracts), March 31, 2013 (mini-symposia), July 31, 2013 (full papers).
☺ Sep 30- Oct 04	12th **International Conference on Parallel Computing Technologies** (PaCT'2013), Saint-Petersburg, Russia. Topics include: new developments, applications, and trends in parallel computing technologies; all technological aspects of the applications of parallel computer systems; high level parallel programming languages and systems; methods and tools for parallel solution of large-scale problems; languages, environments and software tools supporting parallel processing; teaching parallel processing; etc. Deadline for submissions: February 11, 2013 (full papers), February 25, 2013 (extended abstracts).
♦ November	ACM SIGAda Annual International Conference on High Integrity Language Technology (HILT'2013), Pittsburgh, Pennsylvania, USA.
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

Preliminary Call for Participation
Ada Developer Room at FOSDEM 2013
3 February 2013, Brussels, Belgium

Organized by Ada-Belgium
in cooperation with Ada-Europe

FOSDEM[1], the Free and Open source Software Developers' European Meeting, is a free two-day annual event organized in Brussels, Belgium, that offers open source communities a place to meet, share ideas and collaborate. It is renowned for being highly developer-oriented and brings together 5000+ geeks from all over the world. The 2013 edition takes place on Saturday 2 and Sunday 3 February, 2013.

For the 4th time, Ada-Belgium[2] organizes a series of presentations related to Ada and Free Software in a s.c. Developer Room. The "Ada DevRoom" at FOSDEM 2013 is held on the second day of the event, i.e. on Sunday 3 February. The program offers introductory presentations on the Ada programming language, including the recently published new Ada 2012 ISO standard, as well as more specialised presentations on focused topics. We also provide time for discussion and interaction, and organize the by now famous "Adaists dinner" on Saturday evening...

More details are available on the Ada at FOSDEM 2013 web-page, such as the full list with abstracts of presentations, biographies of speakers, and the concrete schedule. For the latest information at any time, contact <Dirk.Craeynest@cs.kuleuven.be>, or see:

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/13/130203-fosdem.html

1 https://fosdem.org/2013
2 http://www.cs.kuleuven.be/~dirk/ada-belgium

16th International Real-Time Ada Workshop (IRTAW 2013)

17-19 April 2013 – Kings Manor, York, England

http://www.cs.york.ac.uk/~andy/IRTAW2013/

Call for Papers

Since the late Eighties the **International Real-Time Ada Workshop** series has provided a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions, and has attracted participation from key members of the research, user, and implementer communities worldwide. Recent IRTAW meetings have significantly contributed to the Ada 2005 and Ada 2012 standards, 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, the goals of IRTAW-16 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 2012 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-16** 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.

Submission requirements

Position papers should not exceed ten pages in typical IEEE conference layout, excluding code inserts. All accepted papers will appear, in their final form, in the Workshop Proceedings, which will be published as a special issue of Ada Letters (ACM Press). Selected papers will also appear in the Ada User Journal.

Please submit position papers, in PDF format, to the Program Chair by e-mail: alan.burns@york.ac.uk

Important Dates

Receipt of Position Paper: 1 February 2013	Final Copy of Paper: 1 April 2013
Notification of Acceptance: 1 March 2013	Workshop Date: 17-19 April 2013

THE UNIVERSITY *of York*

Ada europe

Call for Participation

18th International Conference on Reliable Software Technologies

Ada-Europe 2013

10-14 June 2013, Berlin, Germany

http://www.ada-europe.org/conference2013

Förderverein Ada Deutschland e. V.

Special Interest Group Ada of the German Informatics Society

Conference and Program Co-Chairs

Hubert B. Keller
Karlsruhe Institute of Technology
hubert.keller@kit.edu

Erhard Plödereder
University of Stuttgart
ploedere@iste.uni-stuttgart.de

Tutorial Chair

Jürgen Mottok
Regensburg University of Applied Sciences
Juergen.Mottok@hs-regensburg.de

Industrial Chair

Jørgen Bundgaard
Ada in Denmark
jb@ada-dk.org

Exhibition Chair

Peter Dencker
ETAS GmbH
peter.dencker@etas.com

Publicity Chair

Dirk Craeynest
Ada-Belgium & KU Leuven
Dirk.Craeynest@cs.kuleuven.be

Local Chair

Raúl Rojas
FU Berlin
Raul.Rojas@fu-berlin.de

Local Organizer

Christine Harms
christine.harms@ccha.de

In cooperation with
ACM SIGAda, SIGBED, SIGPLAN

General Information

The 18th International Conference on Reliable Software Technologies – Ada-Europe 2013 will take place in Berlin, Germany. Following its traditional style, the conference will span a full week, including, from Tuesday to Thursday, three days of parallel scientific, technical and industrial sessions, along with parallel tutorials and workshops on Monday and Friday.

Topics

The conference has successfully established itself as an international forum for providers, practitioners and researchers into reliable software technologies across many programming languages. The conference presentations will illustrate current work in the theory and practice of the design, development and maintenance of long-lived, high-quality software systems for a variety of application domains. The program will allow ample time for keynotes, Q&A sessions, panel discussions and social events. Participants will include practitioners and researchers representing industry, academia and government organizations active in the promotion and development of reliable software technologies.
To mark the completion of the **Ada 2012** standard revision process, contributions were sought that discuss experiences with the revised language.

Proceedings

The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer, and will be available at the start of the conference.
The conference is ranked class A in the CORE ranking, is among the top quarter of CiteSeerX Venue Impact Factor, and listed in DBLP, SCOPUS and the Web of Science Conference Proceedings Citation index, among others.

Awards

Ada-Europe will offer honorary awards for the best regular paper and the best presentation.

Program Committee

Ted Baker, US National Science Foundation, USA
Johann Blieberger, Technische Universität Wien, Austria
Bernd Burgstaller, Yonsei University, Korea
Alan Burns, University of York, UK
Rod Chapman, Altran Praxis Ltd, UK
Dirk Craeynest, Ada-Belgium & KU Leuven, Belgium
Juan A. de la Puente, Universidad Politécnica de Madrid, Spain
Franco Gasperoni, AdaCore, France
Michael González Harbour, Universidad de Cantabria, Spain
Xavier Grave, Centre National de la Recherche, France
Christoph Grein, Ada Germany, Germany
J. Javier Gutiérrez, Universidad de Cantabria, Spain
Peter Hermann, Universität Stuttgart, Germany
Jérôme Hugues, ISAE Toulouse, France
Pascal Leroy, Google, Switzerland
Albert Llemosí, Universitat de les Illes Balears, Spain
Kristina Lundqvist, Mälardalen University, Sweden
Franco Mazzanti, ISTI-CNR Pisa, Italy
John McCormick, University of Northern Iowa, USA
Stephen Michell, Maurya Software, Canada
Luís Miguel Pinho, CISTER Research Centre/ISEP, Portugal
Jürgen Mottok, Regensburg University of Applied Sciences, Germany
Manfred Nagl, RWTH Aachen University, Germany
Laurent Pautet, Telecom ParisTech, France
Jorge Real, Universitat Politécnica de València, Spain
Jean-Pierre Rosen, Adalog, France
José Ruiz, AdaCore, France
Ed Schonberg, AdaCore, USA
Tucker Taft, AdaCore, USA
Theodor Tempelmeier, Univ. of Applied Sciences Rosenheim, Germany
Elena Troubitsyna, Åbo Akademi University, Finland
Tullio Vardanega, Università di Padova, Italy
Juan Zamorano, Universidad Politécnica de Madrid, Spain

Industrial Committee

Jørgen Bundgaard, Rambøll Danmark, Denmark
Jacob Sparre Andersen, JSA, Denmark
Jamie Ayre, AdaCore, France
Ian Broster, Rapita Systems, UK
Rod Chapman, Altran Praxis Ltd, UK
Dirk Craeynest, Ada-Belgium & KU Leuven, Belgium
Michael Friess, AdaCore, France
Ismael Lafoz, Airbus Military, Spain
Ahlan Marriott, White-Elephant GmbH, Switzerland
Steen Ulrik Palm, Terma, Denmark
Paolo Panaroni, Intecs, Italy
Paul Parkinson, Wind River, UK
Ana Isabel Rodríguez, GMV, Spain
Jean-Pierre Rosen, Adalog, France
Alok Srivastava, TASC Inc, USA
Claus Stellwag, Elektrobit AG, Germany
Jean-Loup Terraillon, European Space Agency, The Netherlands
Rod White, MBDA, UK

Call for Exhibitors

The commercial exhibition will span the three days of the main conference. Vendors and providers of software products and services should contact the *Exhibition Chair* for information and for allowing suitable planning of the exhibition space and time.

Grant for Reduced Student Fees

A limited number of sponsored grants for reduced fees is expected to be available for students who would like to attend the conference or tutorials. Contact a *Conference Co-Chair* for details.

Location

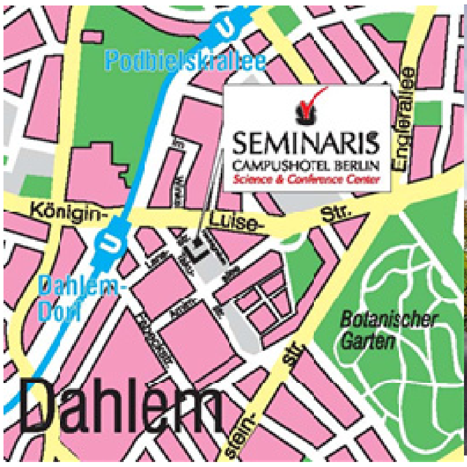
The conference will take place in Berlin, city of many charms. The most difficult activity for tourists is the search for a piece of the Berlin Wall. It has all but disappeared in the vibrant capital of Germany. Berlin has over 170 museums, making the city one of the world's prime locations for first-rate historical art collections, cultural exhibitions, and museums of science and technology. The theatres of performing arts offer programs from opera to musicals, from drama to comedy, from classics to ultra-modern. For a calendar and up-to-date information on events browse under http://www.berlin.de

The conference dinner will take place at the Botanischer Garten of the Freie Universität Berlin.

The conference site is the Seminaris Conference Hotel.

The hotel represents a modern life style and aims to suit the demands of visitors to conferences, business meetings, and cultural programs in Berlin and its surroundings. Each of the 186 first-class rooms are air-conditioned, with bath or shower/WC, TV, minibar, large desk, safe, phone, and high-speed internet access. Named the Dahlem Cube, the conference center in the shape of a glass cube with its 2600 sqm of conference space is a masterpiece of modern architecture by Helmut Jahn, Chicago. The hotel is located in the suburb Dahlem at Takustraße 39, 14195 Berlin, and connected to the center of Berlin by subway transportation of about 30 minutes. Details are found under http://www.seminaris.de/berlin.

A block of rooms at reduced prices has been reserved at the hotel. Attendees are asked to stay at the conference hotel. Details will become available with the publication of the registration form.



For travelling to Berlin by plane, see http://www.berlin-airport.de (Airport Tegel).

ACM SIGAda Annual International Conference

High Integrity Language Technology HILT 2013

Call for Technical Contributions

Developing and Certifying Critical Software

Pittsburgh, Pennsylvania, USA
Fall of 2013 [Mid Oct to early Dec]

Sponsored by ACM SIGAda
`SIGAda.HILT2013@acm.org`
`http://www.sigada.org/conf/hilt2013`

SUMMARY

High integrity software must not only meet correctness and performance criteria but also satisfy stringent safety and/or security demands, typically entailing certification against a relevant standard. A significant factor affecting whether and how such requirements are met is the chosen language technology and its supporting tools: not just the programming language(s) but also languages for expressing specifications, program properties, domain models, and other attributes of the software or overall system.

HILT 2013 will provide a forum for experts from academia/research, industry, and government to present the latest findings in designing, implementing, and using language technology for high integrity software. To this end we are soliciting technical papers, experience reports (including experience in teaching), and tutorial proposals on a broad range of relevant topics.

POSSIBLE TOPICS INCLUDE BUT ARE NOT LIMITED TO:

- New developments in formal methods
- Multicore and high integrity systems
- Object-Oriented Programming in high integrity systems
- High-integrity languages (e.g., SPARK)
- Use of high reliability profiles such as Ravenscar
- Use of language subsets (e.g., MISRA C, MISRA C++)
- Software safety standards (e.g., DO-178B and DO-178C)
- Typed/Proof-Carrying Intermediate Languages
- Contract-based programming (e.g., Ada 2012)
- Model-based development for critical systems
- Specification languages (e.g., Z)
- Annotation languages (e.g., JML)
- Teaching high integrity development
- Case studies of high integrity systems
- Real-time networking/quality of service guarantees
- Analysis, testing, and validation
- Static and dynamic analysis of code
- System Architecture and Design including Service-Oriented Architecture and Agile Development
- Information Assurance
- Security and the Common Criteria / Common Evaluation Methodology
- Architecture design languages (e.g., AADL)
- Fault tolerance and recovery

KINDS OF TECHNICAL CONTRIBUTIONS

TECHNICAL ARTICLES present significant results in research, practice, or education. Articles are typically 10-20 pages in length. These papers will be double-blind refereed and published in the Conference Proceedings and in ACM Ada Letters. The Proceedings will be entered into the widely consulted ACM Digital Library accessible online to university campuses, ACM's 100,000 members, and the software community.

EXTENDED ABSTRACTS discuss current work for which early submission of a full paper may be premature. If your abstract is accepted, a full paper is required and will appear in the proceedings. Extended abstracts will be double-blind refereed. In 5 pages or less, clearly state the work's contribution, its relationship with previous work by you and others (with bibliographic references), results to date, and future directions.

EXPERIENCE REPORTS present timely results and "lessons learned". Submit a 1-2 page description of the project and the key points of interest. Descriptions will be published in the final program or proceedings, but a paper will not be required.

PANEL SESSIONS gather groups of experts on particular topics. Panelists present their views and then exchange views with each other and the audience. Panel proposals should be 1-2 pages in length, identifying the topic, coordinator, and potential panelists.

INDUSTRIAL PRESENTATIONS Authors of industrial presentations are invited to submit a short overview (at least 1 page in size) of the proposed presentation and, if selected, a subsequent abstract for a 30-minute talk. The authors of accepted presentations will be invited to submit corresponding articles for ACM Ada Letters.

WORKSHOPS are focused sessions that allow knowledgeable professionals to explore issues, exchange views, and perhaps produce a report on a particular subject. Workshop proposals, up to 5 pages in length, will be selected based on their applicability to the conference and potential for attracting participants.

TUTORIALS can address a broad spectrum of topics relevant to the conference theme. Submissions will be evaluated based on applicability, suitability for presentation in tutorial format, and presenter's expertise. Tutorial proposals should include the expected level of experience of participants, an abstract or outline, the qualifications of the instructor(s), and the length of the tutorial (half day or full day).

HOW TO SUBMIT: Send in Word, PDF, or text format:

Submission	***Deadline***	***Send to***
Technical articles, extended abstracts, experience reports, panel session proposals, or workshop proposals	**June 29, 2013**	**Tucker Taft,** Program Chair `taft@adacore.com`
Industrial presentation proposals	**August 1, 2013** (overview) **September 30, 2013** (abstract)	
Tutorial proposals	**June 29, 2013**	**John McCormick,** Tutorials Chair `mccormick@cs.uni.edu`

At least one author is required to register and make a presentation at the conference.

FURTHER INFORMATION

CONFERENCE GRANTS FOR EDUCATORS: The ACM SIGAda Conference Grants program is designed to help educators introduce, strengthen, and expand the use of Ada and related technologies in school, college, and university curricula. The Conference welcomes a grant application from anyone whose goals meet this description. The benefits include full conference registration with proceedings and registration costs for 2 days of conference tutorials/workshops. Partial travel funding is also available from AdaCore to faculty and students from GNAT Academic Program member institutions, which can be combined with conference grants. For more details visit the conference web site or contact **Prof. Michael B. Feldman** (`MFeldman@gwu.edu`)

OUTSTANDING STUDENT PAPER AWARD: An award will be given to the student author(s) of the paper selected by the program committee as the outstanding student contribution to the conference.

SPONSORS AND EXHIBITORS: Please contact **Greg Gicca** (`gicca@adacore.com`) to learn the benefits of becoming a sponsor and/or exhibitor at HILT 2013.

***IMPORTANT INFORMATION FOR NON-US SUBMITTERS*:** International registrants should be particularly aware and careful about visa requirements, and should plan travel well in advance. Visit the conference website for detailed information pertaining to visas.

ANY QUESTIONS?

Please send email to `SIGAda.HILT2013@acm.org`, or contact the Conference Chair (**Jeff Boleng**, `jlboleng@SEI.CMU.EDU`), SIGAda's Vice-Chair for Meetings and Conferences (**Alok Srivastava**, `alok.srivastava@tasc.com`), or SIGAda's Chair (**Ricky E. Sward**, `rsward@mitre.org`).

Try and Beat Me!

The Ada Way Rematch.

In 2010 Ada-Europe launched a student programming contest to build a software simulator of a football match. Full sources of the winning submission are published at: www.ada-europe.org/AdaWay.

Do you think you can do better? If so, we're inviting you to make a submission that attempts to improve over the reference implementation under any of the evaluation criteria listed on the Ada Way page.

If you think you're up for the challenge, please visit: www.ada-europe.org/AdaWay

Rationale for Ada 2012: 5 Iterators, Pools, etc.

John Barnes

John Barnes Informatics, 11 Albert Road, Caversham, Reading RG4 7AN, UK; Tel: +44 118 947 4125; email: jgpb@jbinfo.demon.co.uk

Abstract

This paper describes various improvements in a number of general areas in Ada 2012.

There are some minor but perhaps surprising changes concerning matters such as the placement of pragmas and labels.

There are important new features regarding indexing and accessing largely introduced to simplify iterating over containers.

There are also a number of additional Restrictions identifiers many related to the introduction of aspect specifications.

The functionality of access types and storage management is made more flexible by the introduction of subpools.

Finally, a number of minor additions and corrections are made to a range of topics such as generics.

Keywords: rationale, Ada 2012.

1 Overview of changes

The areas mentioned in this paper are not specifically mentioned in the WG9 guidance document [1] other than under the request to remedy shortcomings and improve the functionality of access types and dynamic storage management.

The following Ada Issues cover the relevant changes and are described in detail in this paper.

6	Nominal subtypes for all names
71	Class-wide operations for formal subprograms
95	Address of intrinsic subprograms
100	Placement of pragmas
111	Subpools, allocators & control of finalization
119	Package Calendar, Daylight Saving Time and UTC_Offset
123	Composability of equality
139	Syntactic sugar for access, containers & iterators
148	Accessibility of anonymous access stand-alone objects
149	Access type conversion and membership
152	Restriction No_Anonymous_Allocators
163	Pragmas in place of null
173	Testing of tags representing abstract types
179	Labels at end of a sequence of statements
189	Restriction No_Standard_Allocators_After_Elaboration
190	Global storage pool control
193	Alignment of allocators
212	Accessors and iterators for Ada.Containers
241	Aspect-related restrictions
242	No_Implementation_Units restriction
246	Restrictions No_Implementation_Identifiers and Profile No_Implementation_Extensions
252	Questions on subpools
253	Accessibility of allocators for anonymous access of an object
255	User-defined iterators and quantified expressions
272	Pragma and attribute restrictions
292	Terminology: indexable type is confusing

These changes can be grouped as follows.

First there are some minor changes to elementary matters such as the placement of pragmas, labels and null statements (100, 163, 179).

An important addition is the introduction of more user-friendly mechanisms for iterating over structures such as arrays and containers (139, 212, 255, 292).

Further flexibility for storage management is provided by the introduction of subpools of storage pools (111, 190, 252). A number of issues concerning anonymous access types and allocators are also resolved (148, 149, 193, 253).

A number of new Restrictions identifiers have been added. They include No_Standard_Allocators_After_Elaboration, No_Anonymous_Allocators, No_Implementation_Units, and No_Implementation_Identifiers. A blanket new profile covering a number of restrictions, No_Implementation_Extensions, is also added (152, 189, 241, 242, 246, 272).

Finally, there are a number of minor unrelated improvements. Four are actually classed as binding interpretations and so apply to Ada 2005 as well; they concern nominal subtypes (6), address of intrinsic subprograms (95), time in the package Calendar (119), and class wide operations on formal generic subprograms (71). The other miscellaneous issues concern the composability of equality (123), and tags of abstract types (173).

2 Position of pragmas and labels

It is surprising that basic stuff such as where one can place a pragma should be the subject of discussion thirty years after Ada became an ANSI standard.

However, there is a real problem in this area which one could imagine might have led to headlines in the Wall Street Journal and Financial Times such as

Collapse of NY Stock Market because of Safety Fears in Avionic Applications after Discovery that Ada is Illegal

Indeed, it seems that the package Ada in Ada 2005 might be illegal. This surprising conclusion was triggered by the consideration of

```
task type TT is
   pragma Priority(12);
end TT;
```

The rules in Ada 83, Ada 95 and Ada 2005 concerning the position of pragmas say

Pragmas are only allowed at the following places in a program:

- After a semicolon delimiter, but not within a formal part or discriminant part.
- At any place where the syntax rules allow a construct defined by a syntactic category whose name ends with "declaration", "statement", "clause", or "alternative"; or one of the syntactic categories variant or exception_handler; but not in place of such a construct. Also at any place where a compilation_unit would be allowed.

Now the syntax for task_definition in Ada 2005 is

```
task_definition ::=
   {task_item}
[private
   {task_item}]
end [task_identifier]
```

There are at least two problems. The key one here is that the list of categories in the rule does not include "item". The other concerns the words "not in place of". It seems that the intent was that if at least one instance of the construct is required (as in a sequence of statements) then the pragma cannot be given in place of a single statement. So it looks as if the task type TT is not legal.

It has probably been permitted because task_item itself splits down into aspect_clause or entry_declaration and they seem to be allowed. But if none is present then we cannot tell which category is permitted!

Note rather scarily that the package Ada is given as

```
package Ada is
   pragma Pure(Ada);
end Ada;
```

and the same problem applies.

The entities in a package specification are of the category basic_declarative_item and again although it splits down into things ending _clause or _declaration we don't know which.

The fear concerning package Ada made one member of the ARG concerned that the sky might be falling in. Of course, we don't ever have to submit a package Ada in our file (on punched cards, paper tape or whatever media we are using). The package Ada is just in the mind of the compiler so that it behaves as if she were declared. The same applies to Standard. They are sort of synthesized and not actually declared.

Anyway, the upshot is that in Ada 2012, the description of the placement of pragmas is corrected by adding "item" to the list and clarifying the meaning of not in place of.

A further discussion considered sequences of statements. In a structure such as an if statement the syntax is

```
if_statement ::=
  if condition then
    sequence_of_statements
  ...
```

where

sequence_of_statements ::= statement {statement}

The important point is that a sequence_of_statements must have at least one statement. Moreover, the rules for placing pragmas in Ada 2005 do not allow a pragma in place of a construct so we cannot write

```
if B then
   pragma Assert( ... );   -- illegal in Ada 2005
else ...
```

but have to include at least one statement (such as a null statement) by writing perhaps

```
if B then
   pragma Assert( ... ); null;
else ...
```

or

```
if B then
   null; pragma Assert( ... );
else ...
```

On reflection this seemed irritating so the rules for the placement of pragmas are further amended to include another bullet

- In place of a statement in a sequence_of_statements

A useful note on a language definition principle is added to the AARM which is that if all pragmas are treated as unrecognized then a program should remain legal.

Incidentally, there are other places in the language where at least one item is required such as in a component list. Again if we don't want any components we have to write a null component as in

```
type Nothing is
  record
    null;
  end record;
```

One might have thought that we could similarly now allow one to write

```
type T is
  record
    pragma Page;
  end record;
```

Indeed, it might have been thought that we could simply say that in general a pragma can be given "in place of" an entity. But this doesn't work in some cases. For example, an asynchronous select statement can take the form of a series of statements in its triggering alternative thus

```
select
  S1( ... );
  S2( ... );
  S3( ... );
then abort
  ...
end select;
```

Now the call of S1 is the triggering statement and has a different status to S2 and S3. It would be very confusing to be able to replace the call of S1 by a pragma. So such generalization was dismissed as leading to trouble.

The final topic in this vein concerns the position of labels. This was triggered by the consideration of the problem of quitting one iteration of a loop if it proves unsuccessful and then trying the next iteration. As described in the Introduction this can be done by writing

```
for I in Some_Range loop
  ...
  if not OK then goto End_Of_Loop; end if;
  ...           -- lots of other code
<<End_Of_Loop>> null;         -- try another iteration
end loop;
```

Of course, maybe we should avoid the goto and write

```
for I in Some_Range loop
  ...
  if OK then
    ...         -- lots of other code
  end if;
                -- try another iteration
end loop;
```

At first sight the latter structure looks nicer. However, if the "lots of other code" encounters several situations which mean that the iteration has to be abandoned then we quickly get a deeply nested structure which is not easy to understand and becomes heavily indented.

Much consideration was given to the introduction of a continue statement but it was felt that this would obscure the existence of the transfer of control. Although the goto may be deprecated as obscure, the corresponding obvious label in its aggressive double angle brackets is a strong clue to the existence of the transfer of control.

In the end it was decided that the only sensible improvement was to remove the need for the null statement at the end of the loop.

This is achieved by changing the syntax for a sequence of statements to

sequence_of_statements ::= statement {statement} {label}

and adding a rule to the effect that if one or more labels end a sequence of statements then an implicit null statement is inserted after the labels. So the loop example can now be written as

```
for I in Some_Range loop
  ...
  if not OK then goto End_Of_Loop; end if;
  ...           -- lots of other code
<<End_Of_Loop>>               -- try another iteration
end loop;
```

More generally we can write

```
if B then
  S1; S2; <<My_Label>>
end if;
```

as well as giving the null explicitly thus

```
if B then
    S1; S2; <<My_Label>> null;
end if;
```

but we still cannot write

```
if B then
  <<My_Label>>                -- illegal
end if;
```

since a sequence of statements must still include at least one statement. Of course, we could never jump to such a label anyway since control cannot be transferred into a structure.

3 Iteration

Iteration and subprogram calls are in some sense the twin cornerstones of programming. We are all familiar with the ubiquitous nature of statements such as

```
for I in A'Range loop
  A(I) := 0;
end loop;
```

which in one form or another exist in all (normal) programming languages.

The detail of giving the precise description of the iteration and the indexing is really a violation of abstraction by revealing unnecessary detail. All we want to say is "assign zero to each element of the set A".

However, although it's not too much of a hassle with arrays, the introduction of containers revealed that detailed iteration could be very heavy-handed. Thus, as mentioned in the Introduction, suppose we are dealing with a list, perhaps a list of the type Twin declared as

```
type Twin is
  record
    P, Q: Integer;
  end record;
```

To manipulate every element of the list in Ada 2005, we have to write something like

```
C := The_List.First;        -- C declared as of type Cursor
loop
  exit when C = No_Element;
  E := Element(C);          -- E is of type Twin
  if Is_Prime(E.P) then
    Replace_Element(The_List, C, (E.P, E.Q + X));
  end if;
  C := Next(C);
end loop;
```

This reveals the gory details of the iterative process whereas all we want to say is "add X to the component Q for all members of the list whose component P is prime".

There is another way in Ada 2005 and that is to use the procedure Iterate. In that case the details of what we are doing have to be placed in a distinct subprogram called perhaps Do_It. Thus we can write

```
declare
  procedure Do_It(C: in Cursor) is
  begin
    E := Element(C);          -- E is of type Twin
    if Is_Prime(E.P) then
      Replace_Element(The_List, C, (E.P, E.Q + X));
    end if;
  end Do_It;
begin
  The_List.Iterate(Do_It'Access);
end;
```

This avoids the fine detail of calling First and Next but uses what some consider to be a heavy infrastructure.

However, in Ada 2012 we can simply say

```
for E of The_List loop
  if Is_Prime(E.P) then
    E.Q := E.Q + X;
  end if;
end loop;
```

Not only is this just five lines of text rather than nine or eleven, the key point is that the possibility of making various errors of detail is completely removed.

The mechanisms by which this magic abstraction is achieved are somewhat laborious and it is anticipated that users will take a cookbook approach (show us how to do it, but please don't explain why – after all, this is the approach taken with boiling an egg, we can do it without deep knowledge of the theory of coagulation of protein material).

We will start by looking at the process using arrays. Rather than

```
for I in A'Range loop
  if A(I) /= 0 then
    A(I) := A(I) + 1;
  end if;
end loop;
```

we can write

```
for E of A loop
  if E /= 0 then
    E := E + 1;
  end if;
end loop;
```

In the case of a two-dimensional array, instead of

```
for I in AA'Range(1) loop
  for J in AA'Range(2) loop
    A(I, J) := 0.0;
  end loop;
end loop;
```

we can write

```
for EE of AA loop
  EE := 0.0;
end loop;
```

In Ada 2005 (and indeed in Ada 95 and Ada 83), the syntax for a loop is given by

```
loop_statement ::= [loop_statement_identifier :]
                   [iteration_scheme] loop
                     sequence_of_statements
                   end loop [loop_identifier] ;

iteration_scheme ::= while condition
                     | for loop_parameter_specification

loop_parameter_specification ::= defining_identifier in
                   [reverse] discrete_subtype_definition
```

This is all quite familiar. In Ada 2012, the syntax for loop_statement remains the same but iteration_scheme is extended to give

```
iteration_scheme ::= while condition
                     | for loop_parameter_specification
                     | for iterator_specification
```

Thus the new form iterator_specification is introduced which is

```
iterator_specification ::=
          defining_identifier in [reverse] iterator_name
        | defining_identifier [: subtype_indication] of
                         [reverse] iterable_name
```

The first production defines a *generalized iterator* whereas the second defines an *array component iterator* or a *container element iterator*. For the moment we will just consider the second production which has **of** rather than **in**. The *iterable*_name can refer to an array or a container. Suppose it is an array such as A or AA in the examples above.

We note that we can optionally give the subtype of the loop parameter. Suppose that the type A is given as

```
type A is array (index) of Integer;
```

then the subtype of the loop parameter (E in the example) if not given will just be that of the component which in this case is simply Integer. If we do give the subtype of the loop parameter then it must cover that of the component. This could be useful with tagged types.

Note carefully that the loop parameter does not have the type of the index of the array as in the traditional loop but has the type of the component of the array. So on each iteration it denotes a component of the array. It iterates over all the components of the array as expected. If **reverse** is not specified then the components are traversed in ascending index order whereas if **reverse** is specified then the order is descending. In the case of a multidimensional array then the index of the last dimension varies fastest matching the behaviour of AA in the expanded traditional version as shown (and which incidentally is the order used in streaming). However, if the array has convention Fortran then it is the index of the first dimension that varies fastest both in the case of the loop and in streaming.

There are other obvious rules. If the array A or AA is constant then the loop parameter E or EE is also constant. So it all works much as expected. But do note carefully the use of the reserved word **of** (rather than **is**) which distinguishes this kind of iteration from the traditional form using an index.

As another array example suppose we have the following

```
type Artwin is array (1 .. N) of Twin;

The_Array: Artwin;
```

which is similar to the list example above. In the traditional way we might write

```
for K in Artwin'Range loop
  if Is_Prime(The_Array(K).P) then
    The_Array(K).Q := The_Array(K).Q + X;
  end if;
end loop;
```

Using the new notation this can be simplified to

```
for E: Twin of The_Array loop
  if Is_Prime(E.P) then
    E.Q := E.Q + X;
  end if;
end loop;
```

where we have added the subtype Twin to clarify the situation. Similarly, in the simple list example we could write

```
for E: Twin of The_List loop
  if Is_Prime(E.P) then
    E.Q := E.Q + X;
  end if;
end loop;
```

Note the beautiful similarity between these two examples. The only lexical difference is that The_Array is replaced by The_List showing that arrays and containers can be treated equivalently.

We now have to consider how the above can be considered as behaving like the original text which involves C of type Cursor, and subprograms First, No_Element, Element, Replace_Element and Next.

This magic is performed by several new features. One is a generic package whose specification is

```
generic
  type Cursor;
  with function Has_Element(Position: Cursor)
                                          return Boolean;
package Ada.Iterator_Interfaces is
  pragma Pure(Iterator_Interfaces);

  type Forward_Iterator is limited interface;
  function First(Object: Forward_Iterator)
                              return Cursor is abstract;
  function Next(Object: Forward_Iterator,
           Position: Cursor) return Cursor is abstract;

  type Reversible_Iterator is limited interface and
                                      Forward_Iterator;
  function Last(Object: Reversible_Iterator)
                              return Cursor is abstract;
  function Previous(Object: Reversible_Iterator;
           Position: Cursor) return Cursor is abstract;
end Ada.Iterator_Interfaces;
```

This generic package is used by the container packages such as Ada.Containers.Doubly_Linked_Lists. Its actual parameters corresponding to the formal parameters Cursor and Has_Element come from the container which includes an instantiation of Ada.Iterator_Interfaces. The instantiation then exports the various required types and functions. Thus in outline the relevant part of the list container now looks like

```
with Ada.Iterator_Interfaces;
generic
  type Element_Type is private;
  with function "=" (Left, Right: Element_Type)
                                  return Boolean is <>;
package Ada.Containers.Doubly_Linked_Lists is
  ...
  type List is tagged private ...
  ...
  type Cursor is private;
  ...
  function Has_Element(Position: Cursor)
                                          return Boolean;
  package List_Iterator_Interfaces is
     new Ada.Iterator_Interfaces(Cursor, Has_Element);
  ...
  ...
end Ada.Containers.Doubly_Linked_Lists;
```

The entities exported from the generic package Ada.Iterator_Interfaces are the two interfaces

Forward_Iterator and Reversible_Iterator. The interface Forward_Iterator has functions First and Next whereas the Reversible_Iterator (which is itself descended from Forward_Iterator) has functions First and Next inherited from Forward_Iterator plus additional functions Last and Previous.

Note carefully that a Forward_Iterator can only go forward but a Reversible_Iterator can go both forward and backward. Hence it is reversible and not Reverse_Iterator.

The container packages also contain some new functions which return objects of the type Reversible_Iterator'Class or Forward_Iterator'Class. In the case of the list container they are

```
function Iterate(Container: in List) return
        List_Iterator_Interfaces.Reversible_Iterator'Class;
function Iterate(Container: in List;
                        Start: in Cursor) return
        List_Iterator_Interfaces.Reversible_Iterator'Class;
```

These are new functions and are not to be confused with the existing procedures Iterate and Reverse_Iterate which enable a subprogram to be applied to every element of the list but are somewhat cumbersome to use as shown earlier. The function Iterate with only one parameter is used for iterating over the whole list whereas that with two parameters iterates starting with the cursor value equal to Start.

Now suppose that the list container is instantiated with the type Twin followed by the declaration of a list

```
package Twin_Lists is
  new Ada.Containers.Doubly_Linked_Lists
                        (Element_Type => Twin);
The_List: Twin_Lists.List;
```

So we have now declared The_List which is a list of elements of the type Twin. Suppose we want to do something to every element of the list. As we have seen we might write

```
for E: Twin of The_List loop
  ...              -- do something to E
end loop;
```

However, it might be wise at this point to introduce the other from of iterator_specification which is

defining_identifier **in** [**reverse**] *iterator*_name

This defines a generalized iterator and uses the traditional **in** rather than **of** used in the new array component and container element iterators. Using this generalized form we can write

```
for C in The_List.Iterate loop
  ...              -- do something via cursor C
end loop;
```

In the body of the loop we manipulate the elements using cursors in a familiar way. The reader might wonder why there are these two styles, one using **is** and the other using **of**. The answer is that the generalized iterator is more flexible; for example it does not need to iterate over the whole structure. If we write

```
for C in The_List.Iterate(S) loop
```

then the loop starts with the cursor value equal to S; this is using the version of the function Iterate with two parameters. On the other hand, the array component and container element iterators are more succinct where applicable and are the only from of these new iterators that can be used with arrays.

The generalized iterators for the list container use reversible iterators because the functions Iterate return a value of the type Reversible_Iterator'Class. The equivalent code generated uses the functions First and Next exported from List_Iterator_Interfaces created by the instantiation of Ada.Iterator_Interfaces with the actual parameters The_List.Cursor and The_List.Has_Element. The code then behaves much as if it were (see paragraph 13/3 of subclause 5.5.2 of the RM)

```
C: The_List.Cursor;
E: Twin;
F: Forward_Iterator'Class := The_List.Iterate;
...
C := F.First;
loop
  exit when not The_List.Has_Element(C);
  E := The_List.Element(C);
  ...             -- do something to E
  C := F.Next(C);
end loop;
```

Of course, the user does not need to know all this in order to use the construction. Note that the functions First and Next used here (which operate on the class Forward_Iterator and are inherited by the class Reversible_Iterator) are not to be confused with the existing functions First and Next which act on the List and Cursor respectively. The existing functions are retained for compatibility and for use in complex situations.

It should also be noted that the initialization of F is legal since the result returned by Iterate is a value of Reversible_Iterator'Class and this is a subclass of Forward_Iterator'Class.

If we had written

```
for C in reverse The_List.Iterate loop
  ...             -- do something via cursor C
end loop;
```

then the notional code would have been similar but have used the functions Last and Previous rather than First and Next.

Another point is that the function call F.First will deliver the very first cursor value if we had written The_List.Iterate but the value S if we had written The_List.Iterate(S). Remember that we are dealing with interfaces so there is nothing weird here; the two functions Iterate return different types in the class and these have different functions First so the notional generated code calls different functions.

If we use the form

```
for E: Twin of The_List loop
  ...              -- do something to E
end loop;
```

then the generated code is essentially the same. However, since we have not explicitly mentioned an iterator, a default one has to be used. This is given by one of several new aspects of the type List which actually now is

```
type List is tagged private
  with Constant_Indexing => Constant_Reference,
       Variable_Indexing => Reference,
       Default_Iterator => Iterate,
       Iterator_Element => Element_Type;
```

The aspect we need at the moment is the one called Default_Iterator which as we see has the value Iterate (this is the one without the extra parameter). So the iterator F is initialized with this default value and once more we get

```
C: The_List.Cursor;
E: Twin;
F: Forward_Iterator'Class := The_List.Iterate;
...
```

The use of the other aspects will be explained in a moment.

Lists, vectors and ordered maps and sets can be iterated in both directions. They all have procedures Reverse_Iterate as well as Iterate and the two new functions Iterate return a value of Reversible_Iterator'Class.

However, it might be recalled that the notion of iterating in either direction makes no sense in the case of hashed maps and hashed sets. Consequently, there is no procedure Reverse_Iterate for hashed maps and hashed sets and there is only one new function Iterate which (in the case of hashed maps) is

```
function Iterate(Container: in Map) return
      Map_Iterator_Interfaces.Forward_Iterator'Class;
```

and we note that this function returns a value of Forward_Iterator'Class rather than Reversible_Iterator'Class as in the case of lists, vectors, ordered maps, and ordered sets.

Naturally, we cannot put **reverse** in an iterator over hashed maps and hashed sets nor can we give a starting value. So the following are both illegal

```
for C in The_Hash_Map.Iterate(S) loop   -- illegal

for E of reverse The_Hash_Map loop      -- illegal
```

The above should have given the reader a fair understanding of the mechanisms involved in setting up the loops using the new iterator forms. We now turn to considering the bodies of the loops, that is the code marked "*do something via cursor C* " or "*do something to E* ".

In the Ada 2005 example we wrote

```
if Is_Prime(E.P) then
  Replace_Element(The_List, C, (E.P, E.Q + X));
end if;
```

It is somewhat tedious having to write Replace_Element when using a container whereas in the case of an array we might directly write

```
if Is_Prime(A(I).P) then
  A(I).Q := A(I).Q + X;
end if;
```

The trouble is that Replace_Element copies the whole new element whereas in the array example we just update the one component. This doesn't matter too much in a case where the components are small such as Twin but if they were giant records it would clearly be a problem. To overcome this Ada 2005 includes a procedure Update_Element thus

```
procedure Update_Element(Container: in out List;
                               Position: in Cursor;
                  Process: not null access procedure
                          (Element: in out Element_Type));
```

To use this we have to write a procedure Do_It say thus

```
procedure Do_It(E: in out Twin) is
begin
  E.Q := E.Q + X;
end Do_It;
```

and then

```
if Is_Prime(E.P) then
  Update_Element(The_List, C, Do_It'Access);
end if;
```

This works fine because E is passed by reference and no giant copying occurs. However, the downside is that the distinct procedure Do_It has to be written so that the overall text is something like

```
declare
  procedure Do_It(E: in out Twin) is
  begin
    E.Q := E.Q + X;
  end Do_It;
begin
  if Is_Prime(E.P) then
    Update_Element(The_List, C, Do_It'Access);
  end if;
end;
```

which is a bit tedious.

But of course, the text in the body of Do_It is precisely what we want to say. Using the historic concepts of left and right hand values, the problem is that The_List(C).Element cannot be used as a left hand value by writing for example

```
The_List(C).Element.Q := ...
```

The problem is overcome in Ada 2012 using a little more magic by the introduction of generalized reference types and various aspects. In particular we find that the containers now include a type Reference_Type and a function Reference which in the case of the list containers are

```
type Reference_Type
    (Element: not null access Element_Type) is private
  with Implicit_Dereference => Element;

function Reference(Container: aliased in out List;
            Position: in Cursor) return Reference_Type;
```

Note the aspect Implicit_Dereference applied to the type Reference_Type with discriminant Element.

There is also a type Constant_Reference_Type and a function Constant_Reference for use when the context demands read-only access.

The alert reader will note the inclusion of **aliased** for the parameter Container of the function Reference. As discussed in the paper on Structure and Visibility, this ensures that the parameter is passed by reference (it always is for tagged types anyway); it also permits us to apply 'Access to the parameter Container within the function and to return that access value.

It might be helpful to say a few words about the possible implementation of Reference and Reference_Type although these need not really concern the user.

The important part of the type Reference_Type is its access discriminant. The private part might contain housekeeping stuff but we can ignore that. So in essence it is simply a record with just one component being the access discriminant

```
type Reference_Type
     (E: not null access Element_Type) is null record;
```

and the body of the function might be

```
function Reference(Container: aliased in out List;
            Position: in Cursor) return Reference_Type is
begin
  return (E => Container.Element(Position)'Access);
end Reference;
```

The rules regarding parameters with **aliased** (which we gloss over) ensure that no accessibility problems should arise. Note also that it is important that the discriminant of Reference_Type is an access discriminant since the lifetime of the discriminant is then just that of the return object.

Various aspects are given with the type List which as shown earlier now is

```
type List is tagged private
  with Constant_Indexing => Constant_Reference,
       Variable_Indexing => Reference,
       Default_Iterator => Iterate,
       Iterator_Element => Element_Type;
```

The important aspect here is Variable_Indexing. If this aspect is supplied then in essence the type can be used in a left hand context by invoking the function given as the value of the aspect. In the case of The_List this is the function Reference which returns a value of type Reference_Type. Moreover, this reference type has a discriminant which is of type **access** Element_Type and the aspect Implicit_Dereference with value Element and so gives direct access to the value of type Element.

We can now by stages transform the raw text. So using the cursor form we can start with

```
for C in The_List.Iterator loop
  if Is_Prime(The_List.Reference(C).Element.all.P) then
    The_List.Reference(C).Element.all.Q :=
      The_List.Reference(C).Element.all.Q + X;
  end if;
end loop;
```

This is the full blooded version even down to using **all**.

Omitting the **all** and using the dereferencing with the aspect Implicit_Dereference we can omit the mention of the discriminant Element to give

```
for C in The_List.Iterator loop
  if Is_Prime(The_List.Reference(C).P) then
    The_List.Reference(C).Q :=
      The_List.Reference(C).Q + X;
  end if;
end loop;
```

Remember that Reference is a function with two parameters. It might be clearer to write this without prefix notation which gives

```
for C in Iterator(The_List) loop
  if Is_Prime(Reference(The_List, C).P) then
    Reference(The_List, C).Q :=
      Reference(The_List, C).Q + X;
  end if;
end loop;
```

Now because the aspect Variable_Indexing for the type List has value Reference, the explicit calls of Reference can be omitted to give

```
for C in The_List.Iterator loop
  if Is_Prime(The_List(C).P) then
    The_List(C).Q := The_List(C).Q + X;
  end if;
end loop;
```

It should now be clear that the cursor C is simply acting as an index into The_List. We can compare this text with

```
for C in The_Array'Range loop
  if Is_Prime(The_Array(C).P) then
    The_Array(C).Q := The_Array(C).Q + X;
  end if;
end loop;
```

which shows that 'Range is analogous to .Iterator.

Finally, to convert to the element form using E we just replace The_List(C) by E to give

```
for E of The_List loop
  if Is_Prime(E.P) then
    E.Q := E.Q + X;
  end if;
end loop;
```

The reader might like to consider the transformations in the reverse direction to see how the final succinct form transforms to the expanded form using the various aspects. This is indeed what the compiler has to do.

This underlying technique which transforms the sequence of statements of the container element iterator can be used quite generally. For example, we might not want to iterate over the whole container but just manipulate a particular element given by a cursor C. Rather than calling Update_Element with another subprogram Do_Something, we can write

```
The_List.Reference(C).Q := ...
```

or simply

```
The_List(C).Q := ...
```

Moreover, although the various aspects were introduced into Ada 2012 primarily to simplify the use of containers they can be used quite generally.

The reader may feel that these new features violate the general ideas of a language with simple building blocks. However, it should be remembered that even the traditional form of loop such as

```
for Index in T range L to U loop
   ...                        -- statements
end loop;
```

is really simply a shorthand for

```
declare
   Index: T;
begin
   if L <= U then
      Index := L;
      loop
         ...                  -- statements
         exit when Index = U;
         Index := T'Succ(Index);
      end loop;
   end if;
end;
```

Without such shorthand, programming would be very tedious and very prone to errors. The features described in this section are simply a further step to make programming safer and simpler.

Further examples of the use of these new features with containers will be given in a later paper dedicated to containers.

The mechanisms discussed above rely on a number of new aspects, a summary of which follows and might be found useful. It is largely based on extracts from the RM.

Dereferencing

The following aspect may be specified for a discriminated type T.

Implicit_Dereference This aspect is specified by a name that denotes an access discriminant of the type T.

A type with a specified Implicit_Dereference aspect is a *reference type*. The Implicit_Dereference aspect is inherited by descendants of type T if not overridden.

A generalized_reference denotes the object or subprogram designated by the discriminant of the reference object.

Indexing

The following aspects may be specified for a tagged type T.

Constant_Indexing This aspect is specified by a name that denotes one or more functions declared immediately within the same declaration list in which T is declared. All such functions shall have at least two parameters, the first of which is of type T or T'Class, or is an access-to-constant parameter with designated type T or T'Class.

Variable_Indexing This aspect is specified by a name that denotes one or more functions declared immediately within the same declaration list in which T is declared. All such functions shall have at least two parameters, the first of which is of type T or T'Class, or is an access parameter with designated type T or T'Class. All such functions shall have a return type that is a reference type, whose reference discriminant is of an access-to-variable type.

These aspects are inherited by descendants of T (including T'Class). The aspects shall not be overridden, but the functions they denote may be.

An *indexable container type* is a tagged type with at least one of the aspects Constant_Indexing or Variable_Indexing specified.

An important difference between Constant_Indexing and Variable_Indexing is that the functions for variable indexing must return a reference type so that it can be used in left hand contexts such as the destination of an assignment. Note that, in both cases, the name can denote several overloaded functions; this is useful, for example, with maps to allow indexing both with cursors and with keys.

Both Constant_Indexing and Variable_Indexing can be provided since the constant one might be more efficient whereas the variable one is necessary in left hand contexts. But we are not obliged to give both, just Variable_Indexing might be enough for some applications.

Iterating

An iterator type is a type descended from the Forward_Iterator interface.

The following aspects may be specified for an indexable container type T.

Default_Iterator This aspect is specified by a name that denotes exactly one function declared immediately within the same declaration list in which T is declared, whose first parameter is of type T or T'Class or an access parameter whose designated type is type T or T'Class, whose other parameters, if any, have default expressions, and whose result type is an iterator type. This function is the *default iterator function* for T.

Iterator_Element This aspect is specified by a name that denotes a subtype. This is the *default element subtype* for T.

These aspects are inherited by descendants of type T (including T'Class).

An *iterable container type* is an indexable container type with specified Default_Iterator and Iterator_Element aspects.

The Constant_Indexing and Variable_Indexing aspects (if any) of an iterable container type T shall denote exactly one function with the following properties:

- the result type of the function is covered by the default element type of T or is a reference type with an access discriminant designating a type covered by the default element type of T;
- the type of the second parameter of the function covers the default cursor type for T;
- if there are more than two parameters, the additional parameters all have default expressions.

These functions (if any) are the *default indexing function*s for T.

The reader might care to check that the aspects used in the examples above match these definitions and are used correctly. Note for example that the Default_Iterator and Iterator_Element aspects are only needed if we use the **of** form of iteration (and both are needed in that case, giving one without the other would be foolish).

This section has largely been about the use of iterators with loop statements. However, there is one other use of them and that is with quantified expressions which are also new to Ada 2012. Quantified expressions were discussed in some detail in the paper on Expressions so all we need here is to consider a few examples which should clarify the use of iterators.

Instead of

```
B := (for all K in A'Range => A(K) = 0);
```

which assigns true to B if every component of the array A has value 0, we can instead write

```
B := (for all E of A  => E = 0);
```

Similarly, instead of

```
B := (for some K in A'Range => A(K) = 0);
```

which assigns true to B if some component of the array A has value 0, we can instead write

```
B := (for some E of A => E = 0);
```

In the case of a multidimensional array, instead of

```
B := (for all I in AA'Range(1) =>
        (for all J in AA'Range(2) => AA(I, J) = 0));
```

we can write

```
B := (for all E of AA => E = 0);
```

which iterates over all elements of the array AA however many dimensions it has.

We can also use these forms with the list example. Suppose we are interested in checking whether some element of the list has a prime component P. We can write

```
B := (for some E of The_List => Is_Prime(E.P));
```

or perhaps

```
B := (for some C in The_List.Iterator =>
                              Is_Prime(The_List(C).P));
```

which uses the explicit iterator form.

4 Access types and storage pools

A significant change in Ada 2005 was the introduction of anonymous access types. It is believed that the motivation was to remove the feeling that Ada 95 was unnecessarily pedantic in requiring the introduction of lots of named access types whereas in languages such as C one can just place a star on the identifier of the type being referenced in order to introduce a pointer type.

However, anonymous access types raised more complex accessibility check problems which did not arise with named access types. Most of these problems were resolved in the definition of Ada 2005 but one remained concerning stand-alone objects of anonymous access types. Interestingly, such stand-alone objects were added to Ada 2005 late in the development process; perhaps hastily as it turned out.

In Ada 2005, local stand-alone objects take the accessibility level of the master in which they are declared.

Consider an attempt to use a local stand-alone object in an algorithm to reverse a list. We assume that the list comprises nodes of the following type

```
type Node is
  record
    ...
    Next: access Node;
  end record;
```

and we write

```
function Reverse(List: access Node)
                                    return access Node is
  Result: access Node := null;
  This_Node: access Node := List;
  Next_Node: access Node := null;
begin
  while This_Node /= null loop
    Next_Node := This_Node.Next;
    This_Node.Next := Result;    -- access failure in 2005
    Result := This_Node;
    This_Node := Next_Node;
  end loop;
  return Result;                  --access failure in 2005
end Reverse;
```

This uses the obvious algorithm of working down the list and rebuilding it. However, in Ada 2005 there are two accessibility failures associated with the variable Result. The assignment to This_Node.Next fails because Result might be referring to something local and we cannot assign that to a node of the list since the list itself lies outside the scope of Reverse_List. Similarly, attempting to return the value in Result fails.

The problem with returning a result can sometimes be solved by using an extended return statement as illustrated in [2]. But this is not a general remedy. The problem is solved in Ada 2012 by treating stand-alone access objects rather like access parameters so that they carry the accessibility of the last value assigned to them as part of their value.

Another reason for introducing anonymous access types in Ada 2005 was to reduce the need for explicit type conversions (note that anonymous access types naturally have no name to use in an explicit conversion). However, it turns out that in practice it is convenient to use anonymous access types in some contexts (such as the component Next of type Node) but in other contexts we might find it logical to use a named access type such as

```
type List is access Node;
```

In Ada 2005, explicit conversions are often required from anonymous access types to named access types and this has been considered to be irritating. Accordingly, the rule has been changed in Ada 2012 to say that an explicit conversion is only required if the conversion could fail.

This relaxation covers both accessibility checks and tag checks. For example we might have

```
type Class_Acc is access T'Class;          -- named type
type Rec is
  record
    Comp: access T'Class;                  -- anon type
  end record;

R: Rec;
```

and then some code somewhere

```
Z: Class_Acc;
...
Z := R.Comp;                          -- OK in Ada 2012
```

The conversion from the anonymous type of Comp to the named type Class_Acc of Z on the assignment to Z cannot fail and so does not require an explicit conversion whereas it did in Ada 2005. Incidentally, the example uses a component Comp rather than a stand-alone object to avoid confusion arising from the special properties of stand-alone objects just discussed.

With regard to tag checks, if it is statically known that the designated type of the anonymous access type is covered by the designated type of the named access type then there is no need for a tag check and so an explicit conversion is not required.

It will be recalled that there is a fictitious type known as *universal_access* (much as *universal_integer*, *root_Integer* and so on). For example, the literal **null** is of this universal type. Moreover, there is a function "=" used to compare *universal_access* values. Permitting implicit conversions requires the introduction of a preference rule for the equality operator of the universal type. Suppose we have

```
type A is access Integer;
R, S: access Integer;
...
if R = S then
```

Now since we can do an implicit conversion from the anonymous access type of R and S to the type A, there is confusion as to whether the comparison uses the equality operator of the type *universal_access* or that of the type A. Accordingly, there is a preference rule that states that in the case of ambiguity there is a preference for equality of the type *universal_access*. Similar preference rules already apply to *root_integer* and *root_real*.

A related topic concerns membership tests which were described in the paper on Expressions.

If we want to ensure that a conversion from perhaps Integer to Index will work and not raise Constraint_Error we can write

```
subtype Index is Integer range 1 .. 20;
I: Index;
K: Integer;
...
if K in Index then
  I := Index(K);            -- bound to work
else
  ...                       -- remedial action
end if;
```

This is much neater than attempting the conversion and then handling Constraint_Error.

However, in Ada 2005, there is no similar facility for testing to see whether an access type conversion would fail. So membership tests in Ada 2012 are extended to permit such a test. So if we have

```
type A is access T1;
X: A;
...
type Rec is
  record
    Comp: access T2;
  end record;

R: Rec;
Y: access T2;
```

we can write

```
if R.Comp in A then
  X := A(R.Comp)            -- conversion bound to work
else ...
```

The membership test will return true if the type T1 covers T2 and the accessibility rules are satisfied so that the conversion is bound to work. Note that the converted expression (R.Comp in this case) can be an access parameter or a stand-alone access object such as Y.

We now turn to consider various features concerning allocation and storage pools.

It will be recalled that if we write our own storage pools then we have to declare a pool type derived from the type Root_Storage_Pool in the package System.Storage_Pools. So we might write

```
package My_Pools is
  type Pond(Size: Storage_Count) is
                  new Root_Storage_Pool with private;
  ...
```

where the discriminant gives the size of the pool. We then have to provide procedures Allocate and Deallocate for our own pool type Pond corresponding to those for Root_Storage_Pool. The procedures Allocate and Deallocate both have four parameters. For example, the procedure Allocate is

```
procedure Allocate(Pool: in out Root_Storage_Pool;
                   Storage_Address: out Address;
                   Size_In_Storage_Elements,
                   Alignment: in Storage_Count) is abstract;
```

When we declare our own Allocate we do not have to use the same names for the formal parameters. So we might more simply write

```
procedure Allocate(Pool: in out Pond;
                        Addr: out Address;
                        SISE: in Storage_Count;
                        Align: in Storage_Count);
```

As well as Allocate and Deallocate we also have to write a function Storage_Size and procedures Initialize and Finalize. However, the key procedures are Allocate and Deallocate which give the algorithms for determining how the storage in the pool is manipulated.

Two parameters of Allocate give the size and alignment of the space to be allocated. However, it is possible that the particular algorithm devised might need to know the worst case values in determining an appropriate strategy. The attribute Max_Size_In_Storage_Elements gives the worst case for the storage size in Ada 2005 but there is no corresponding attribute for the worst case alignment.

This is overcome in Ada 2012 by the provision of the attribute Max_Alignment_For_Allocation. There are various reasons for possibly requiring a different alignment to that expected. For example, the raw objects might simply be byte aligned but the algorithm might decide to append dope or monitoring information which is integer aligned.

The collector of Ada curiosities might remember that Max_Size_In_Storage_Elements is the attribute with most characters in Ada 2005 (28 of which 4 are underlines). Curiously, Max_Alignment_For_Allocation also has 28 characters of which only 3 are underlines.

There are problems with anonymous access types and allocation. Consider

```
package P is
  procedure Proc(X: access Integer);
end P;

with P;
procedure Try_This is
begin
  P.Proc(new Integer'(10));
end Try_This;
```

The procedure Proc has an access parameter X and the call of Proc in Try_This does an allocation with the literal 10. Where does it go? Which pool? Can we do Unchecked_Deallocation? There are special rules for allocators of anonymous access types which aim to answer such questions. The pool is "created at the point of the allocator" and so on.

But various problems arise. An important one is that it is not possible to do unchecked deallocation because the access type has no name; this is particularly serious with library level anonymous access types. An example of such a type might be that of the component Next if the record type Node discussed earlier had been declared at library level.

Consequently, it was concluded that it is best to use named access types if allocation is to be performed. We can always convert to an anonymous type if desired after the allocation has been performed.

In order to avoid encountering such problems a new restriction identifier is introduced. So writing

```
pragma Restrictions(No_Anonymous_Allocators);
```

prevents allocators of anonymous access types and so makes the call of the procedure Proc in the procedure Try_This illegal.

Many long-lived control programs have a start-up phase in which various storage structures are established and which is then followed by the production phase in which various restrictions may be imposed. Ada 2012 has a number of features that enable this to be organized and monitored.

One such feature is the new restriction

```
pragma Restrictions(No_Standard_Allocators_
                                        After_Elaboration);
```

This specifies that an allocator using a standard storage pool shall not occur within a parameterless library subprogram or within the statements of a task body. In essence this means that all such allocation must occur during library unit elaboration. Storage_Error is raised if allocation occurs afterwards.

However, it is expected that systems will permit some use of user-defined storage pools. To enable the writers of such pools to monitor their use some additional functions are added to the package Task_Identification so that it now takes the form

```
package Ada.Task_Identification is
  ...
```

```
      type Task_Id is private;
      ...
      function Current_Task return Task_Id;
      function Environment_Task return  Task_Id;
      procedure Abort_Task(T: in Task_Id);

      function Is_Terminated(T: Task_Id) return Boolean;
      function Is_Callable(T: Task_Id) return Boolean;
      function Activation_Is_Complete(T: Task_Id)
                                        return Boolean;
   private
      ...
   end Ada.Task_Identification;
```

The new function Environment_Task returns the identification of the environment task. The function Activation_Is_Complete returns true if the task concerned has finished activation. Moreover, if Activation_Is_Complete is applied to the environment task then it indicates whether all library items of the partition have been elaborated.

A major new facility is the introduction of subpools. This is an extensive subject so we give only an overview. The general idea is that one wants to manage heaps with different lifetimes. It is often the case that an access type is declared at library level but various groups of objects of the type are declared and so could be reclaimed at a more nested level. This is done by splitting a pool into separately reclaimable subpools. This is far safer and often cheaper than trying to associate lifetimes with individual objects.

A new child package of System.Storage_Pools is declared thus

```
   package System.Storage_Pools.Subpools is
      pragma Preelaborate(Subpools);

      type Root_Storage_Pool_With_Subpools is
          abstract new Root_Storage_Pool with private;

      type Root_Subpool is
          abstract tagged limited private;

      type Subpool_Handle is
          access all Root_Subpool'Class;
        for Subpool_Handle'Storage_Size use 0;

      function Create_Subpool
           (Pool: in out Root_Storage_Pool_With_Subpools)
                 return not null Subpool_Handle is
                                          abstract;

      function Pool_of_Subpool
          (Subpool: not null Subpool_Handle) return
          access Root_Storage_Pool_With_Subpools'Class;

      procedure Set_Pool_of_Subpool
                         (Subpool: not null Subpool_Handle;
       To: in out Root_Storage_Pool_With_Subpools'Class);

      procedure Allocate_From_Subpool(
          Pool: in out Root_Storage_Pool_With_Subpools;
          Storage_Address: out Address;
          Size_In_Storage_Elements: in Storage_Count;
          Alignment: in Storage_Count;
          Subpool: in not null Subpool_Handle) is abstract
          with Pre'Class =>
                Pool_of_Subpool(Subpool) = Pool'Access;

      procedure Deallocate_Subpool(
          Pool: in out Root_Storage_Pool_With_Subpools;
          Subpool: in out Subpool_Handle) is abstract
          with Pre'Class =>
                Pool_of_Subpool(Subpool) = Pool'Access;

      function Default_Subpool_for_Pool
          (Pool: in out Root_Storage_Pool_With_Subpools)
                   return not null Subpool_Handle;

      overriding
      procedure Allocate(
          Pool: in out Root_Storage_Pool_With_Subpools;
          Storage_Address: out Address;
          Size_In_Storage_Elements: in Storage_Count;
          Alignment: in Storage_Count);

      overriding
      procedure Deallocate( ... ) is null;

      overriding
      function Storage_Size
             (Pool : Root_Storage_Pool_With_Subpools)
           return Storage_Count is
                (Storage_Count'Last);

   private
      ... -- not specified by the language
   end System.Storage_Pools.Subpools;
```

If we wish to declare a storage pool that can have subpools then rather than declare an object of the type Root_Storage_Pool in the package System.Storage_Pools we have to declare an object of the derived type Root_Storage_Pool_With_Subpools declared in the child package.

The type Root_Storage_Pool_With_Subpools inherits operations Allocate, Deallocate and Storage_Size from the parent type. Remember that Allocate and Deallocate are automatically called by the compiled code when items are allocated and deallocated. In the case of subpools we don't need Deallocate to do anything so it is null. The function Storage_Size determines the value of the attribute Storage_Size and is given by a function expression.

Subpools are separately reclaimable parts of a storage pool and are identified and manipulated by objects of the type Subpool_Handle (these are access values). We can create a subpool by a call of Create_Subpool. So we might have (assuming appropriate with and use clauses)

```
   package My_Pools is
      type Pond(Size: Storage_Count) is
      new Root_Storage_Pool_With_Subpools with private;

      subtype My_Handle is Subpool_Handle;
      ...
```

and then

```
   My_Pool: Pond(Size => 1000);
```

```
Puddle: My_Handle := Create_Subpool(My_Pool);
```

The implementation of Create_Subpool should call

```
Set_Pool_Of_Subpool(Puddle, My_Pool);
```

before returning the handle. This enables various checks to be made.

In order to allocate an object of type T from a subpool, we have to use a new form of allocator. But first we must ensure that T is associated with the pool itself. So we might write

```
type T_Ptr is access T;
for T_Ptr'Storage_Pool use My_Pool;
```

And then to allocate an object from the subpool identified by the handle Puddle we write

```
X := new (Puddle) T'( ... );
```

where the subpool handle is given in parentheses following **new**.

Of course we don't have to allocate all such objects from a specified subpool since we can still write

```
Y := new T'( ... );
```

and the object will be allocated from the parent pool My_Pool. It is actually allocated from a default subpool in the parent pool and this is determined by writing a suitable body for the function Default_Subpool_for_Pool and this is called automatically by the allocation mechanism. Note that in effect the whole of the pool is divided into subpools one of which may be the default subpool. If we don't provide an overriding body for Default_Subpool_For_Pool then Program_Error is raised. (Note that this function has a parameter of mode **in out** for reasons that need not bother us.)

The implementation carries out various checks. For example, it will check that a handle refers to a subpool of the correct pool by calling the function Pool_Of_Subpool. Both this function and Set_Pool_Of_Subpool are provided by the Ada implementation and typically do not need to be overridden by the implementor of a particular type derived from Root_Storage_Pool_With_Subpools.

In the case of allocation from a subpool, the procedure Allocate_From_Subpool rather than Allocate is automatically called. Note the precondition to check that all is well.

It will be recalled that for normal storage pools, Deallocate is automatically called from an instance of Unchecked_Deallocation. In the case of subpools the general idea is that we get rid of the whole subpool rather than individual items in it. Accordingly, Deallocate does nothing as mentioned earlier and there is no Deallocate_From_Subpool. Instead we have to write a suitable implementation of Deallocate_Subpool. Note again the precondition to check that the subpool belongs to the pool.

Deallocate_Subpool is called automatically as a consequence of calling the following library procedure

```
with System.Storage_Pools.Subpools;
use System.Storage_Pools.Subpools;
procedure Ada.Unchecked_Deallocate_Subpool(
                              Subpool: in out Subpool_Handle);
```

So when we have finished with the subpool Puddle we can write

```
Unchecked_Dellocate_Subpool(Puddle);
```

and the handle becomes null. Appropriate finalization also takes place.

In summary, the writer of a subpool implementation typically only has to provide Create_Subpool, Allocate_From_Subpool and Deallocate_Subpool since the other subprograms are provided by the Ada implementation of the package System.Storage_Pools.Subpools and can be inherited unchanged.

An example of an implementation will be found in subclause 13.11.6 of the RM. This shows an implementation of a Mark/Release pool in a package MR_Pool. Readers are invited to create variants called perhaps Miss_Pool and Dr_Pool!

Further control over the use of storage pools (nothing to do with subpools) is provided by the ability to define our own default storage pool as mentioned in the Introduction. Thus we can write (and completing our Happy Family of Pools)

```
pragma Default_Storage_Pool(Master_Pool);
```

and then all allocation within the scope of the pragma will be from Master_Pool unless a different specific pool is given for a type. This could be done by using an attribute definition clause thus

```
type Cell_Ptr is access Cell;
for Cell_Ptr'Storage_Pool use Cell_Ptr_Pool;
```

or by using an aspect specification thus

```
type Cell_Ptr is access Cell
  with Storage_Pool => Cell_Ptr_Pool;
```

A pragma Default_Storage_Pool can be overridden by another one so that for example all allocation in a package (and its children) is from another pool.

The default pool can be specified as **null** thus

```
pragma Default_Storage_Pool(null);
```

and this prevents any allocation from standard pools.

Allocation normally occurs from the default pool unless a specific pool has been given for a type. But there are two exceptions, one concerns access parameter allocation and the other concerns coextensions; in these cases allocation uses a pool that depends upon the context.

Thus in the case of the procedure Proc discussed above, a call such as

```
P.Proc(new Integer'(10));
```

might allocate the space in a secret pool created on the fly and that secret pool might be placed on the stack.

Such allocation can be prevented by two more specific restrictions. They are

```
pragma Restriction(No_Access_Parameter_Allocators);
```

and

```
pragma Restriction(No_Coextensions);
```

These two pragmas plus using the restriction Default_Storage_Pool with **null** ensure that all allocation is from user-defined pools.

5 Restrictions

Restrictions provide a valuable way of increasing security. Ada is a rich language and even richer with Ada 2012 and although individual features are straightforward, certain combinations can cause problems.

The new restrictions introduced into Ada 2012 have already been described in this or earlier papers such as the Introduction. However, for convenience here is a complete list giving the annex where appropriate.

The new Restrictions identifiers are

No_Access_Parameter_Allocators	High-Integrity
No_Anonymous_Allocators	High-Integrity
No_Cooextensions	High-Integrity
No_Implementation_Aspect_Specifications	
No_Implementation_Identifiers	
No_Implementation_Units	
No_Specification_Of_Aspect	
No_Standard_Allocators_After_Elaboration	Real-Time
No_Use_Of_Attribute	
No_Use_Of_Pragma	

Some of the new Restrictions identifiers are in the High-Integrity annex. They are

```
pragma Restrictions(No_Access_Parameter_Allocators);

pragma Restrictions(No_Anonymous_Allocators);

pragma Restrictions(No_Coextensions);
```

and these were discussed in the previous section.

In a similar vein there is one new restriction in the Real-Time annex, namely

```
pragma Restrictions(No_Standard_Allocators_
                                After_Elaboration);
```

and this was also discussed in the previous section.

A number of restrictions prevent the use of implementation-defined features. They are

```
pragma Restrictions(No_Implementation_Aspect_
                                     Specifications);

pragma Restrictions(No_Implementation_Identifiers);

pragma Restrictions(No_Implementation_Units);
```

These do not apply to the whole partition but only to the compilation or environment concerned. This helps us to ensure that implementation dependent areas of a program are identified. They were discussed in the Introduction and join similar restrictions No_Implementation_Attributes and No_Implementation_Pragmas introduced in Ada 2005.

The restrictions on implementation-defined aspect specifications, attributes and pragmas are obvious but some clarification of what is meant by the restrictions on units and identifiers might be helpful.

It will be recalled that the predefined packages are Ada, System and Interfaces plus various children. In the so-called standard mode, implementations are not permitted to add their own child packages of Ada but can add grandchildren. Thus an implementation might add an additional container package called perhaps Ada.Containers.Slopbucket. If a program were to use this grandchild then clearly it would be unlikely to be portable to other implementations. Accordingly, giving the restriction No_Implementation_Units prevents such potential difficulties. Similarly, this restriction prevents the use of implementation-defined child units of System and Interfaces.

The restriction No_Implementation_Identifiers is more subtle. It will be recalled that several predefined packages are permitted to add implementation-defined identifiers. They are

Standard, System, Ada.Command_Line, Interfaces.C, Interfaces.C.Strings, Interfaces.C.Pointers, Interfaces.COBOL, and Interfaces.Fortran.

Moreover, the following predefined packages only contain implementation-defined identifiers

Interfaces, System.Machine_Code, Ada.Directories.Information, Ada.Directories.Names, and the packages Implementation nested in the queue containers.

The restriction No_Implementation_Identifiers prevents the use of any of these.

There is a slight subtlety regarding Long_Integer and Long_Float in Standard. The types Integer and Float must be provided. Types such as Short_Integer and Long_Long_Float may be provided but are definitely considered to be implementation-defined and so excluded by the restriction on implementation identifiers. However, Long_Integer and Long_Float should be provided (if the hardware is capable) and so are considered to be predefined and not covered by the restriction. Nevertheless, an implementation on a specialized small machine might not provide them.

Finally, there are restrictions preventing the use of particular facilities

```
pragma Restrictions(No_Specification_Of_Aspect => X);

pragma Restrictions(No_Use_Of_Attribute => X);

pragma Restrictions(No_Use_Of_Pragma => X);
```

where X is the name of a specific aspect, attribute or pragma respectively. They are similar to the restriction No_Dependence introduced in Ada 2005. They apply to a complete partition.

Note that No_Specification_Of_Aspect prevents the specification of an aspect by any means. Remember that some aspects can be specified by an aspect specification or by a pragma or by an attribute definition clause. Thus we mentioned above that a storage pool could be given by an attribute definition clause thus

```
type Cell_Ptr is access Cell;
for Cell_Ptr'Storage_Pool use Cell_Ptr_Pool;
```

or by using an aspect specification thus

```
type Cell_Ptr is access Cell
  with Storage_Pool => Cell_Ptr_Pool;
```

Writing

```
pragma Restrictions(No_Specification_Of_Aspect =>
                                        Storage_Pool);
```

prevents both of these whereas

```
pragma Restrictions(No_Use_Of_Attribute =>
                                        Strorage_Pool);
```

prevents only the first. Naturally, No_Use_Of_Attribute prevents both setting an attribute and using it whereas No_Specification_Of_Aspect prevents just setting it. Thus we might want to use 'Size but prevent setting it.

Similarly

```
pragma Restrictions(No_Specification_Of_Aspect =>
                                                Pack);
```

prevents both

```
type Flags is array (1 .. 8) of Boolean
  with Pack;
```

and

```
type Flags is array (1 .. 8) of Boolean;
pragma Pack(Flags);
```

whereas

```
pragma Restrictions(No_Use_Of_Pragma => Pack);
```

prevents only the latter.

In summary, No_Specification_Of_Aspect does not mean No_Aspect_Specification (which does not exist).

Remember that several restrictions can be given in one pragma, so we might have

```
pragma Restrictions(No_Use_Of_Pragma => P,
                    No_Use_Of_Attribute  => A);
```

As mentioned in the Introduction there is also a new profile No_Implementation_Extensions. This is specified by

```
pragma Profile(No_Implementation_Extensions);
```

and is equivalent to writing

```
pragma Restrictions(
      No_Implementation_Aspect_Specifications,
      No_Implementation_Attributes,
      No_Implementation_Identifiers,
      No_Implementation_Pragmas,
      No_Implementation_Units);
```

thus providing blanket security against writing programs that use language extensions. This profile is defined in the core language. The only other profile defined in Ada 2012 is Ravenscar which was introduced in Ada 2005 and is in the Real-Time systems annex. Remember that the pragma Profile is a configuration pragma.

Finally, those of a recursive nature might note that writing

```
pragma Restrictions(No_Use_Of_Pragma =>
                                        Restrictions);
```

is illegal (this prevents the risk that the compiler might melt down). More curiously, there is not a restriction No_Implementation_Restrictions. This might be because of similar concern regarding what would happen with its recursive use.

6 Miscellanea

A number of improvements do not neatly fit into any other section of these papers and so are lumped together here.

The first four are in fact binding interpretations and thus apply to Ada 2005 as well.

First, nominal subtypes are defined for enumeration literals and attribute references so that all names now have a nominal subtype.

This is clearly a matter for the language lawyer rather than the happy programmer. Consider the following weird example

```
subtype S is Integer range 1 .. 10;
...
case S'Last is
  when 0 =>   --  ????
```

This is clearly nonsense. However, Ada 2005 does not define a nominal subtype for attributes such as S'Last and so we cannot determine whether 0 is allowed as a discrete choice. The language definition is tidied up to cover such cases.

The second gap in Ada 2005 concerns intrinsic subprograms. Remember that intrinsic subprograms are functions such as "+" on the type Integer that only exist in the mind of the compiler. Clearly they have no address. The following is added to the RM:

> The prefix of X'Address shall not statically denote a subprogram that has convention Intrinsic. X'Address raises Program_Error if X denotes a subprogram that has convention Intrinsic.

The dynamic check is needed because of the possibility of passing an intrinsic operation as a generic parameter.

The third of these binding gems concerns the package Ada.Calendar.

The problem is that Calendar.Time is not well-defined when a time zone change occurs as for example when Daylight Saving Time is introduced or removed. Thus operations involving several time values (such as subtraction) might give the "correct" answer or might be an hour adrift. The conclusion reached was simply to admit that it is not defined so the wording is slightly changed.

Another problem with the wording in Ada 2005 is that the sign of the difference between local time and UTC as returned by UTC_Offset is not clearly defined. The sign is clarified so that for example UTC_Offset is negative in the American continent.

There is another problem with the package Calendar which will need to be addressed at some time (probably long after the author is dead). Much effort was exerted in Ada 2005 to cope with leap seconds. These arise because the angular velocity of rotation of the Earth is gradually slowing down. In earlier epochs when measurements of time were not accurate this did not matter. However, we now have atomic clocks and the slowdown is significant so that clocks are adjusted by one second as necessary and these are known as leap seconds.

But leap seconds are under threat. There is a move to suggest that tiny adjustments of one second are not worth the effort and that we should wait until the time is a whole hour wrong. A simple adjustment similar to that with which we are familiar with Daylight Saving changes is all that is needed. In other words we will have a leap hour every now and then. Indeed, if leap seconds occur about once a year as they have done on average since 1972 then a leap hour will be needed sometime in the 37th century. This will probably need to be addressed in Ada 3620 or so.

The final binding interpretation concerns class wide types and generics.

An annoyance was recently discovered concerning the use of the indefinite container packages such as

```
generic
   type Index_Type is range <>;
   type Element_Type(<>) is private;
   with function "=" (Left, Right: Element_Type)
                                   return Boolean is <>;
package Ada.Containers.Indefinite_Vectors is
   ...
```

We can instantiate this with an indefinite type such as String by writing perhaps

```
package String_Vectors is
   new Containers.Indefinite_Vectors(Positive, String);
```

The third actual parameter can be omitted because the predefined operation "=" on the type String exists and does what we want.

Class wide types are another example of indefinite types. Thus we might like to create a vector container whose elements are a mixture of objects of types Circle, Square, Triangle and so on. Assuming these are all descended from the abstract type Object we want to instantiate with the class wide type Object'Class.

However, unlike String, class wide types such as Object'Class do not have a predefined equals. This is annoying since the derived types Circle, Square, and Triangle (being just records) do have a predefined equals.

So we have to write something like

```
function Equal(L, R: Object'Class) is
begin
   return L = R;
end Equal;
```

Note that this will dispatch to the predefined equals of the type of the objects passed as parameters. They both must be of the same type of course; we cannot compare a Circle to a Triangle (anymore than we can compare Thee to a Summer's Day).

So we can now instantiate thus

```
package Object_Vectors is
   new Containers.Indefinite_Vectors(
                        Positive, Object'Class, Equal);
```

Note irritatingly that we cannot write Equal as just "=" because this causes ambiguities.

This is all a bit annoying and so in Ada 2012, the required "=" is automatically created, we do not have to declare Equal, and the instantiation can simply be

```
package Object_Vectors is
   new Containers.Indefinite_Vectors(
                        Positive, Object'Class);
```

This improvement is also a binding interpretation and so applies to Ada 2005 as well.

A more serious matter is the problem of the composability of equality.

In Ada 2005, tagged record types compose but untagged record types do not. If we define a new type (a record type, array type or a derived type) then equality is defined in terms of equality for its various components. However, the behaviour of components which are records is different in Ada 2005 according to whether they are tagged or not. If a component is tagged then the primitive operation is used (which might have been redefined), whereas for an untagged type, predefined equality is used even though it might have been overridden.

Consider

```
type Tagrec is tagged
   record
      X1: Integer;
      X2: Integer;
   end record;
```

```
type Untagrec is
  record
    Y1: Integer;
    Y2: Integer;
  end record;

type Index is range 0 .. 64;

...

function "=" (L, R: Tagrec) return Boolean is
begin
  return L.X1 = R.X1;     -- compare only first component
end;

function "=" (L, R: Untagrec) return Boolean is
begin
  return L.Y1 = R.Y1;     -- compare only first component
end;

function "=" (L, R: Index) return Boolean is
begin
  raise Havoc;
  return False;
end;

...

type Mixed is
  record
    T: Tagrec;
    U: Untagrec;
    Z: Index;
  end record;
```

Here we have a type Mixed whose components are of a tagged record type Tagrec, an untagged record type Untagrec, and an elementary type Index. Moreover, we have redefined equality for these types.

In Ada 2005, the equality for the type Mixed uses the redefined equality for the component T but the predefined equality for U and Z. Thus it compares T.X1, U.Y1 and U.Y2 and does not raise Havoc.

In Ada 83, the predefined equality always emerged for the components of arrays and records. One reason was to avoid confusion if an inconsistency arose between "=", "<" and "<=". Remember that many elementary types and certain array types have predefined "<" as well as "=" and to get the relationship messed up would have been confusing.

However, Ada 95 introduced tagged record types and inheritance of operations became an important feature. So it seemed natural that if a structure (array or record) had components of a tagged type and equality for that tagged type had been redefined then it would be natural to expect that equality for the structure should use the redefined equality. But, fearful of introducing an incompatibility, the rule for untagged record types was left unchanged so that predefined equality reemerges.

On reflection, this difference between tagged and untagged records was surprising and so has been changed in Ada 2012 so that all record types behave the same way and use the primitive operation. This is often called composability of equality so we can say that in Ada 2012, record types always compose for equality. Remember that this only applies to records; components which are of array types and elementary types continue to use predefined equality. So in Ada 2012, equality for Mixed only compares T.X1 and U.Y1 but not U.Y2 and still does not raise Havoc.

Concern for incompatibility and inconsistency has been allayed by a deep analysis of a number of programs. No nasties were revealed and in the only cases where it made a difference it was clear that the original behaviour was in fact wrong.

The final miscellaneum (singular of miscellanea?) concerns tags.

The package Ada.Tags defines various functions operating on tags. For example

```
function Parent_Tag(T: Tag) return Tag;
```

returns the tag of the parent unless the type has no parent in which case it returns No_Tag.

However, in Ada 2005 there is no easy way to test whether a tag corresponds to an abstract type. The key property of abstract types is that we cannot have an object of an abstract type. If we wish to create an object using Generic_Dispatching_Constructor and the tag represents an abstract type then Tag_Error is raised. However, it would be far better to check whether a tag represents an abstract type before using Generic_Dispatching_Constructor.

Moreover, if we have a tag and wish to know whether it represents an abstract type, then in Ada 2005 there is no sensible way to find out. We could attempt to create an object and see if it raises Tag_Error. If it doesn't then we know that it was not abstract but we have also created an object we maybe didn't want; if it does raise Tag_Error then it might or might not have been abstract since there are other reasons for the exception being raised. Either way this is madness.

In Ada 2012, we can test the tag using the new function

```
function Is_Abstract(T: Tag) return Boolean;
```

which is added near the end of the package Ada.Tags just before the declaration of the exception Tag_Error.

References

[1] ISO/IEC JTC1/SC22/WG9 N498 (2009) *Instructions to the Ada Rapporteur Group from SC22/WG9 for Preparation of the Amendment.*

[2] John Barnes (2006) *Programming in Ada 2005*, Addison-Wesley.

Tool Support for Verification of Software Timing and Stack Usage for a DO-178B Level A System

Eng. Felipe Kamei, Eng. Daniela Cristina Carta

Embraer, Sao Jose dos Campos, Brazil; email: felipe.kamei@embraer.com.br, daniela.carta@embraer.com.br

Dr. Ian Broster, Will Lunniss

Rapita Systems Ltd.,York, England; email: ianb@rapitasystems.com, wlunniss@rapitasystems.com

Abstract

This paper describes an approach to obtain worst-case execution times and worst-case stack usage using a method supported by the Rapita Verification Suite (RVS) from Rapita Systems Ltd for a DO-178B level A Flight Control System software application. The approach has been applied in a Research and Development project in Embraer. The paper outlines the technical approach, and the results achieved so far using two successful testing strategies.

Keywords: WCET, timing, Stack, DO178-B.

1 Introduction

For critical aerospace applications, it is necessary to demonstrate that the software complies with timing and memory related non-functional requirements. Compliance to the RTCA DO-178B (Software Considerations in Airborne Systems and Equipment Certification) [1] guidelines is the usual method to achieve approval for software implementing critical functions of aircraft systems for aircraft certification regulations. One of the verification activities required by DO-178B is to obtain the worst-case timing and the stack usage values for the application. This ensures that the implementation of the software is accurate, consistent and compliant with its non-functional requirements. Further, in the real-time software application area, optimizing the usage of resources, such as CPU usage and stack memory usage is a valuable exercise.

The relevant section (section 6.3.4) of DO-178B is entitled “Reviews and Analyses of the Source Code". An approach consisting of a combination of functional, timing and stack usage analysis performed by executing tests on the software running on the hardware may be used to accomplish this objective [2]. In this paper, we describe performing software timing and stack analysis in a R&D project whose main purpose is to generate and exercise processes for the development of critical aircraft systems and software.

The system developed as the POC (Proof of Concept) for the processes proposed is a Flight Control System (FCS). This FCS is designed to provide aircraft flight dynamics control by employing Fly-By-Wire (FBW) technology through the actuation of aerodynamic surfaces in response to crew (pilot and co-pilot) commands. Data processing and the logic of the FCS system are implemented by software, which due to the criticality of the FCS function, is classified as Level A software.

One of the drivers for the R&D project is the use of tools to decrease development cost. The proposed method for timing and stack analysis is aided by a tool, named Rapita Verification Suite (RVS). Tool qualification and the activities that need to be performed are discussed later in the paper.

Section 2 provides an overview of the characteristics of the FCS software application. Section 3 presents an overview of RVS together with the proposed method for timing and stack analysis in DO-178B, and Section 4 shows how the RVS integration was performed with the FCS application. Section 5 shows the results obtained, while Section 6 concludes.

2 FCS Software Application

The software under analysis implements the FCS system logic and data processing. It is written in the C programming language and contains approximately 73,000 lines of code. The hardware chosen to support the application is based on a 7448 PowerPC microprocessor running an operating system compliant with the ARINC 653 standard [3].

The FCS software development complies with DO-178B Level A objectives. Hence the whole set of activities required need to be performed for the FCS application. This includes producing requirements, architecture and design documentation artefacts and the review of said artefacts. It additionally covers verification using inspections and tests, structural coverage analysis using MC/DC criteria, and object code analysis, among others. Initial requirements for the FCS application required a worst-case execution time (WCET) of less than 5 milliseconds, and a maximum stack usage of 20,000 bytes.

3 RVS and combination of analysis and test for timing and stack analysis

RVS is a set of tools for on-target verification of software timing (RapiTime), code coverage (RapiCover) and stack usage (RapiSafeStack, currently a prototype) from Rapita Systems Ltd. RVS provides an automated method of software instrumentation, measurement and worst-case analysis. In this project, the particular features of RVS being used are RapiTime for WCET analysis and RapiSafeStack for worst-case stack analysis.

RapiTime combines execution time measurements with structural code analysis. It takes detailed timing measurements from test cases executed on an embedded target or on a simulator, and then uses source-code analysis to construct the worst-case path, worst-case execution time and other metrics [4].

The method of using the RapiTime tool in this project comprises the following stages:

- Using RapiTime to add instrumentation to the source code, according to a predefined level of instrumentation suitable for each procedure.
- Building the instrumented code with the compiler configured in the same way as it is used in the development activities.
- Using RapiTime to analyze the source code and generate an overall structure of the code and the paths through it.
- Exercising the instrumented application on the representative target using test case scenarios that cover all the added instrumentation points.
- Capturing execution data according to a predefined extraction method.
- Using RapiTime to generate a report based on the execution data results and the code structure.
- Based on this report, comparing the non-functional requirements' expected values with the values obtained.

The most important part of this process is that it is not strictly necessary to explicitly design a test case for the worst-case path because RapiTime will both calculate the worst-case path and report whether that path corresponds to the longest path observed during testing. This leads to a significant reduction in effort since generating a worst-case execution time scenario is a time consuming activity for a complex application.

The process for use of RapiSafeStack is similar to the one used for RapiTime. However, the traces used by RapiSafeStack are different in two ways.

- Instrumentation points are only used for function entry and exits.
- Timestamps are replaced with value of the stack pointer.

The results of RVS automated activities are not verified by other means; therefore tool qualification according to DO-178B for RVS's features is necessary. As RVS does not introduce errors in the application and may only fail to detect an error, it can be qualified as a verification tool (which requires less effort than qualification for a development tool). For that, Rapita Systems can provide a qualification kit for the RapiTime tool, which contains a set of documentation for qualification activities compliant to DO-178B, which then need to be complemented by tool user activities in the user environment. As RapiSafeStack is still a prototype, it does not contain this set of documentation yet.

4 Integration of RVS and FCS application

The steps to prepare and execute the RapiTime and RapiSafeStack processes were performed using the FCS software. The level of instrumentation chosen for timing analysis was such that a measurement point was inserted in each branch, start and end of functions of the code. Source code for the operating system functions was not available, so these calls within the application were treated as "black boxes" to simply measure the execution time of the operating system functions end to end, without analyzing them internally.

For stack analysis, inserting a measurement point at the start and end of each function was sufficient. The application required approximately 13,000 instrumentation points for timing analysis and 152 instrumentation points for stack analysis.

4.1 Testing

The initial intention was to exercise the whole integrated application on the representative target; however, due to project decisions, functional verification and structural coverage analysis of the software was performed only on parts of the internal component level of the FCS application. Functional verification of the whole integrated application was performed at the higher level system process perspective, without covering all the paths of the software. Functional test case scenarios for the whole integrated application were not available in the software development process, so some input arrays had to be generated specifically for the timing and stack analysis. 785 test scenarios were created, varying the 500 parameters of the input array of the application. Since the purpose of this testing activity was only to exercise the paths through the application, the outputs were not evaluated and could be ignored. This method achieved coverage of approximately 77% of the instrumentation points for timing analysis, and 100% of the instrumentation points for stack analysis.

Higher coverage is normally expected for WCET analysis with RapiTime; otherwise justification is needed that untested code cannot lie on the worst-case path. However, devising additional test scenarios would require considerable effort in order to exercise the remainder of the code. This is because some specific internal conditions must be met from an external environment perspective. Out of the application's 73,000 lines of code, about 3000 instrumentation points were not covered.

Figure 1 shows a very simplified view of the FCS Application as a block diagram. The blocks described as "Components tested according to formal process (MC/DC coverage)" represent the components selected as samples to fully exercise the formal process aiming to comply with all DO-178B Level A objectives. For such components, test drivers were already available and timing and stack analysis could be performed considering them as standalone software.

In the initial approach, where the whole application is considered, components described as "Components not feasible (deactivated code)", were not analyzed in this task and their execution time and stack usage contribution were not evaluated. This can happen, for example, because of some constant input replicating the output from an external system still in development or that will not be part of the final solution. This constant input then drives execution such that certain components' functionalities are not used.

Therefore, an alternative approach was taken where the analysis from the system-level functional tests were complemented by exercising some of the application's internal components as standalone software modules. This allowed the test cases prepared for formal functional verification to be reused. This approach exercised internal components of the FCS application using already available test drivers, in the same way as used for functional testing. It was possible to reuse the test cases generated for functional verification of the application, without any additional effort.

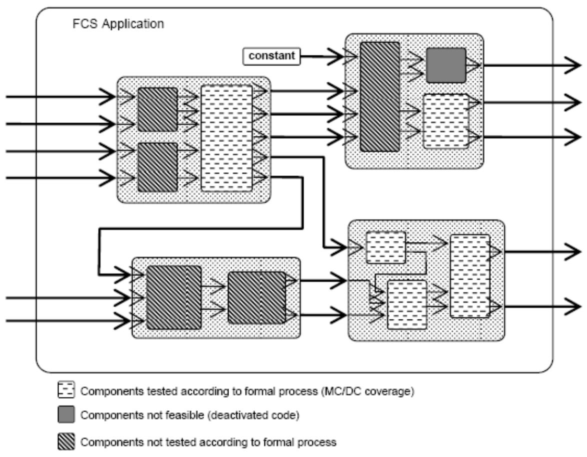

FCS Application
constant
Components tested according to formal process (MC/DC coverage)
Components not feasible (deactivated code)
Components not tested according to formal process

Figure 1 - Different verification approaches in FCS Application components

However, following this approach requires a justification that the timing results from component tests are applicable to system tests. Reasons that they might not match include: components were executed in a different context (i.e. each component executed standalone, rather than as part of the whole application), configurations might be different for both approaches (internal components as standalone software are tested with drivers, which does not happen with the whole integrated application) and the influence of cache memory and the execution of hidden operating system instructions.

To address the applicability of the timing results from these tests, a small, but important, component of the FCS was analyzed using both approaches. This component already had a larger test set of 1200 functional tests which also provided full MC/DC coverage and, in consequence, full branch coverage. When this component was analyzed, the WCET estimate was found to be slightly higher with the isolated, but larger test set, compared to the other approach where this same component was part of the whole application but only partially covered.

4.2 Trace Data Extraction

Finally, an important part in performing RVS tool integration is the trace data extraction from the target. In this project, when the application was complete and integrated on-target, external communication could only be done through a specific aeronautic bus. This was successfully used to transfer the data off the target by writing it through this bus to a PC, using the concept of a queuing service.

For the analysis of the application's internal components using test drivers executed on the target, the trace data extraction method used was writing to a file stored on the target environment, and transferring it to the PC using a normal file transfer protocol.

5 Results

After running and processing data, the WCET value for the whole integrated application was computed as 2.279 ms and the worst-case stack usage was 3416 bytes. For the WCET analysis, 77% coverage was achieved for the instrumentation points with the set of test scenarios used. For stack analysis, 100% coverage of the instrumentation points was achieved.

With the partial timing coverage results, the results confirmed that the FCS was compliant with the required limit on the execution time of 5 ms, and that the stack usage was below the limit of 20,000 bytes. Furthermore, some optimizations to the FCS application with regard to the usage of allocated platform resources could be applied. A 4 ms window was reserved, instead of the previous 5 ms value, which still allows some room for future expansion, but is still well above the obtained value of 2.279 ms.

6 Summary and Next Steps

In summary, the viability of this method of measuring WCET and worst-case stack usage has been successfully shown.

From the project management point of view, this method brings the benefit of obtaining a WCET and stack usage value without the need to generate and demonstrate the test scenario that provokes the WCET and worst-case stack value, which for complex application is a high resource consuming activity. Additionally, once the integration is set up, the method can be repeated with relative ease.

From a certification perspective, the method may be acceptable since it is supported by tool qualification activities and by the same level of rigor as used when performing tests on target for functional verification.

There are still some particular issues to address in the future. Firstly, although tracing via the available platform bus was possible, other more efficient methods to access data externally while the application is running, like a real-time debugger, should also be considered earlier in the design process and possibly before the selection of the target platform.

Second, even though it is not necessary to obtain the test case scenario that exercises exactly the worst-case path, it is necessary to have a test set that exercises most (if not all) the code branches or justify by review that the untested code cannot affect the worst-case execution time. Considering the whole integrated application, such a test set is not always available or practical. However, when it is not available, internal components of the application can be exercised as standalone applications. In this case, it is necessary to evaluate how different hardware conditions affect each approach.

Finally, it is important to note that the usage of black box instrumentation for operating system functions is convenient, but instead of a detailed analysis of the internals of the function, only the highest recorded execution time for the functions is considered, without confirming that their worst-case scenario has been exercised. It would therefore be better to have access to the source code for these functions, so that they can be instrumented or reviewed in order to obtain justification that the OS functions are adequately tested for timing.

Our next steps are to exercise tool qualification of tools considering this method, to analyze the target influences in time measurements and, finally, to implement the process on a software development project in an aircraft program.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 287519, parMERASA.

We acknowledge the support of the EPSRC, via grant EP/F501374/1, through the Engineering Doctorate Centre in Large-Scale Complex IT Systems.

References

[1] RTCA (1992), *Software Consideration in Airborne Systems and Equipment Certification*, RTCA/DO-178B.

[2] RTCA (2001), *Final report for clarification of DO-178B: Software Consideration in Airborne Systems and Equipment Certification*, RTCA/DO-248B.

[3] ARINC 653 Standard (2006), *Avionics Application Software Standard Interface*.

[4] Rapita Systems Ltd (2010), *RapiTime White Paper*. http://www.rapitasystems.com/

Including Hardware/Software Co-design in the ASSERT Model Driven Engineering Process *

Francisco Ferrero, Elena Alaña, Ana Isabel Rodríguez
GMV, E-28760 Madrid, Spain; email: {fferrero,ealana}@gmv.com
Juan Zamorano, Juan A. de la Puente
Universidad Politécnica de Madrid (UPM), E-28040 Madrid, Spain; email: {jzamora,jpuente}@dit.upm.es

Abstract

The ASSERT project defined new software engineering methods and tools for the development of critical embedded real-time systems in the space domain. The ASSERT model-driven engineering process was one of the achievements of the project and is based on the concept of property-preserving model transformations. The key element of this process is that non-functional properties of the software system must be preserved during model transformations. Properties preservation is carried out through model transformations compliant with the Ravenscar Profile and provides a formal basis to the process. In this way, the so-called Ravenscar Computational Model is central to the whole ASSERT process. This paper describes the work done in the HWSWCO study, whose main objective has been to address the integration of the Hardware/Software co-design phase in the ASSERT process. In order to do that, non-functional properties of the software system must also be preserved during hardware synthesis.

Keywords: Ada 2005, Ravenscar profile, Hardware/-Software co-design, real-time systems, high-integrity systems, ORK.

1 Overview

Embedded systems are commonly built by specifying and developing independently the hardware and software subsystems from the system specification phase and their integration is performed lately at the end of the system development. However, current embedded systems are more complex and have shorter time-to-market and thus it is needed to adopt new techniques to reduce time and costs by simplifying the hardware/software integration process. The hardware/software co-design exploits the trade-offs between hardware and software in a system by developing concurrently both systems. The HW/SW co-development approach mainly consists of specifying the system functions (typically in a behavioural form) in a representation that is independent from the underlying execution platform, modelling the system platform describing the hardware architecture of the target platform, partitioning the system into either hardware or software, implementing the hardware and software systems, and scheduling the execution of the tasks to meet any timing constraint.

The ASSERT Project[1] was aimed at defining new software engineering methods and tools for the development of critical embedded real-time systems in the space domain. One of its main achievements is a new model-driven software process, which is based on the concept of property-preserving model transformations [1]. The key element of the ASSERT process is that non-functional properties must be preserved during all phases of model transformations. Therefore, the different views of models must be consistent and thus a a common Component Model (CM) was defined to provide formal basis to the ASSERT process. Moreover, the resulting software architecture must be amenable for response time analysis and thus that CM is compatible with the Ada Ravenscar Profile [2] and is called the Ravenscar Computational Model (RCM). The required real-time behaviour must be guaranteed at execution time and thus the ASSERT execution platform or Virtual Machine (VM) [3] is bound to a GNAT cross-compilation system that only accepts Ravenscar Profile compliant code. In order to ensure compatibility with the Ravenscar Profile, application-level software is built in such a way that all functional code and data is encapsulated into *VM-Level Containers* (VMLC), which are the run-time code entities that operate on top of the VM. Accordingly, VMLC are coded in Ada 2005 [4] restricted by the Ravenscar profile. However, the functional code can be written in Ada, C or C++.

This paper provides the feedback on the research activities carried out in the Hardware-Software co-design discipline and the conclusions extracted from HWSWCO ESA study. The study has investigated the HW/SW co-design phase to integrate this engineering task as part of the ASSERT process and make it compatible with its approach, process and the existing tool-set called TASTE[2].

2 The HW/SW co-development Methodology

The HW/SW co-development methodology is commonly divided into four different phases: co-specification, co-design, co-synthesis and co-validation. The proposed methodology

*This work has been funded by ESA, contract no. 22810/09/NL/JK (HWSWCO).

[1] Automated proof-based system and software engineering for real-time systems, see www.assert-project.net.

[2] The ASSERT Set of Tools for Engineering

starts from an abstract system description based on system behaviour and generates the architecture gradually adding implementation details to the design. In the context of HWSWCO study, the first three phases were investigated to ensure consistency between hardware and software design and verification. Figure 1 shows the proposed methodology that is based on an iterative process compliant with the ASSERT process.

The implementation of co-specification, co-design and co-synthesis phases for this study is supported by a set of standard notations and tools that are included in Figure 1 together with the output of each phase. The HWSWCO process uses the model viewpoints from the ASSERT process that describe the application by means of different model views that separates the data modelling, functional and interface definition in different AADL [5] models that form part of the Platform Independent Model (PIM), according to the MDA guide [6]. The ASSERT Model Transformation (AMT) tool has been implemented in order to support the different transformations to be performed during the co-specification and co-design phases. In this way, the tool processes those model viewpoints, combines them with the description of the platform in the Platform Description Model (PDM), and generates the input to the ASSERT toolset (i.e. Data, Interface and Deployment views).

Co-specification: the system functionality is described independently of the system architecture and constitutes a unified and unbiased representation of the system. The system model encloses the System Logic View and System Platform View following the same model-driven approach proposed in ASSERT.

Co-design: the system components are mapped to processing resources and the feasibility of the partition is then evaluated (which system functions are mapped either to software or hardware). The feasibility analysis is accomplished by performing high-level estimation analysis which provides figures about the use of the platform resources. Those figures will drive the partitioning process although it is necessary calculating the Worst Case Execution Time of the sequential code of system components of the software system and then a schedulability analysis in order to assess partition scheme.

Co-synthesis: the HW and SW systems are synthesized and integrated. In order to do this integration, communication interfaces between the HW and SW systems have to be generated. The usage of a common data representation and a unified system model makes it easier and faster enabling the automatic synthesis of the communication interfaces for communicating both systems.

3 ASSERT Model Transformation

GMV's ASSERT Model Transformation (AMT) tool is central to the HWSWCO approach as it supports different Model-to-Model transformations to be performed during the co-design phase. Figure 2 shows the four basic operations that the tool performs:

1. Imports the ASSERT Data, Functional and Interface views. This function imports them to the AADL project and creates an internal representation of the PIM.
2. Automatically generates the System Partitioned View from the System Model (Logic and Platform views).
3. Transforms the AADL system model into models compatible with the ASSERT model views, i.e., ASSERT Data, Functional, Interface and Deployment Views.
4. Finally, it generates the whole system and loads the ASSERT Concurrency View required by AADS-T to execute the performance analysis on the SW system.

The performance analysis performed during the co-design phase is supported by AADS-T and SCoPE tools, developed by the University of Cantabria. These tools provide the following features:

- Implementing the generation of RCM-compliant SystemC in SCoPE.
- Implementing the estimation of the bus load in SCoPE.
- Modelling the LEON2 processor for SCoPE.
- Generating a SystemC executable model that includes system description.

AADS-T enables the modeling of a subset of AADL including the Behavioral Annex for purposes of implementation and simulation. The starting point of the simulator is a functional AADL specification that is parsed and a model suitable for simulation with SCoPE is produced. SCoPE simulates the SystemC code generated by AADS-T by using a POSIX interface. AADS-T tool was improved in such a way that the generated code is RCM-compliant.

The AMT and AADS-T tools are developed as Eclipse plug-ins integrated with the OSATE AADL editor.

4 Case study

With the goal to exercise the HW/SW co-design methodology, a case study was developed based on a simplified space application for digital image processing. The figure 3 shows the platform used in the HWSWCO case study that is composed of two different processing boards:

FPGA design board: it hosts the HW system that consists of the Image Processing System, formed by the Image Processing Module, Science Data Reporting Module and Control Module.

LEON2 design board: it hosts the SW system that consists of the OBSW. The OBSW receives the start-up command through a serial line and sends commands to control the image processing functions of the HW system.

Moreover, the camera is replaced by a PC that communicates to the FPGA through a serial bus.

Figure 4 shows the main functional blocks of the OBSW and their interaction with the hardware components deployed in the FPGA.

The image processing application performs the following functions:

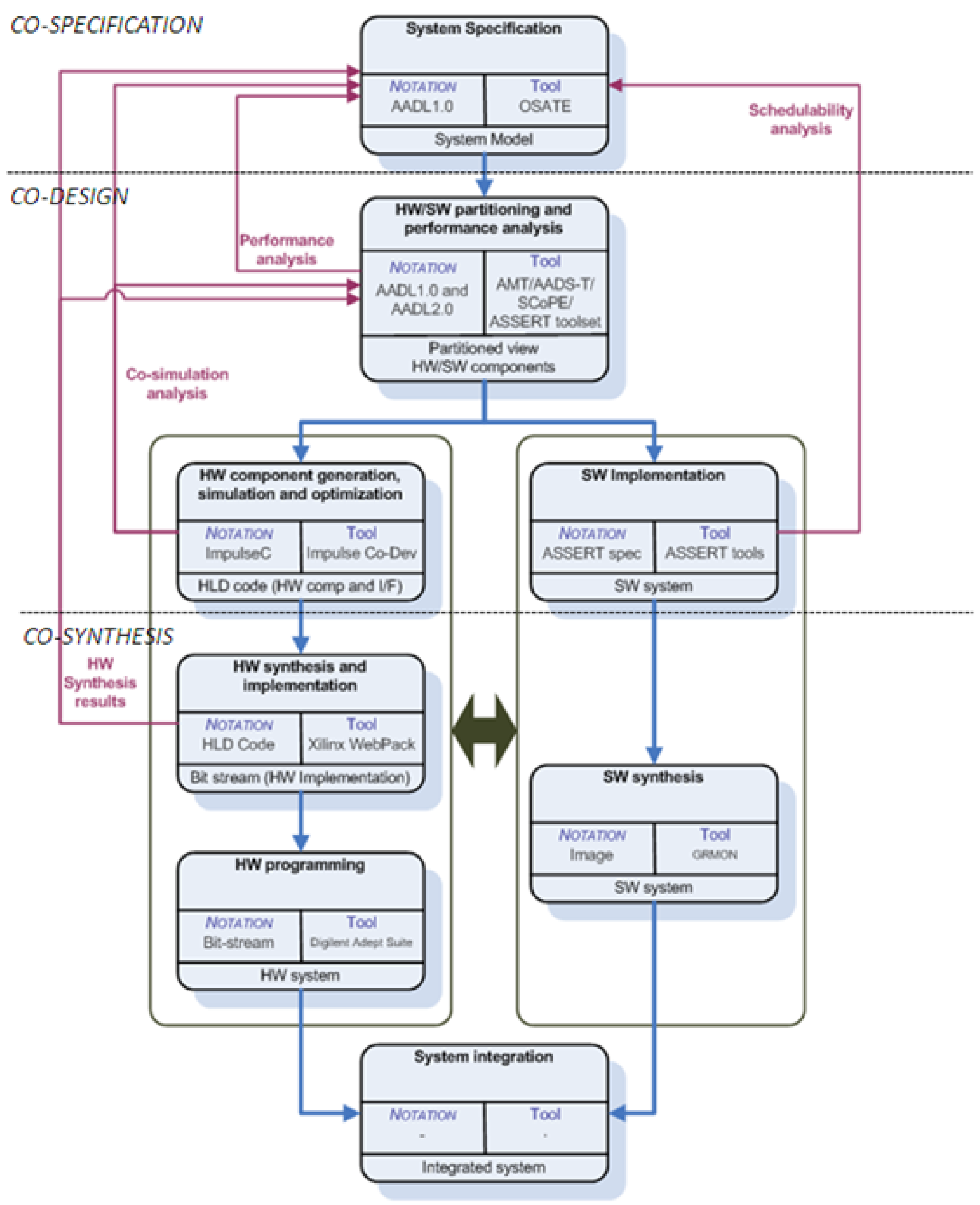


Figure 1: HWSWCO methodology, notations and tools.

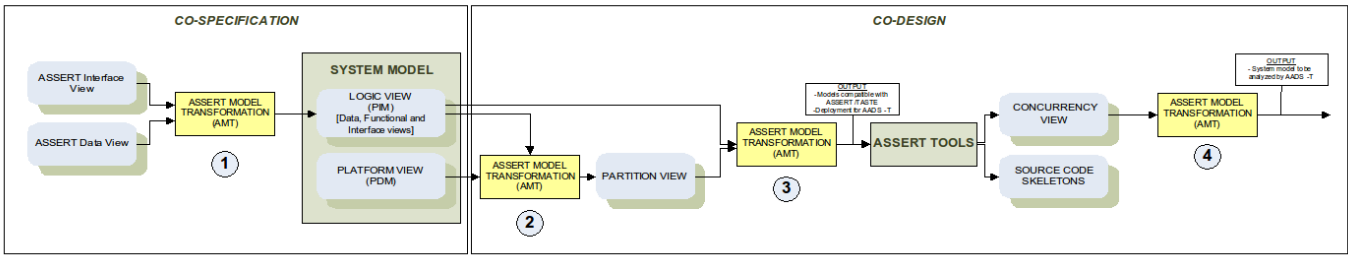

CO-SPECIFICATION
ASSERT Interface View
ASSERT Data View
ASSERT MODEL TRANSFORMATION (AMT)
1
SYSTEM MODEL
LOGIC VIEW (PIM) [Data, Functional and Interface views]
PLATFORM VIEW (PDM)
CO-DESIGN
ASSERT MODEL TRANSFORMATION (AMT)
2
PARTITION VIEW
OUTPUT -Models compatible with ASSERT /TASTE -Deployment for AADS -T
ASSERT MODEL TRANSFORMATION (AMT)
3
ASSERT TOOLS
CONCURRENCY VIEW
SOURCE CODE SKELETONS
OUTPUT - System model to be analyzed by AADS -T
ASSERT MODEL TRANSFORMATION (AMT)
4

Figure 2: ASSERT model transformation tool operation.

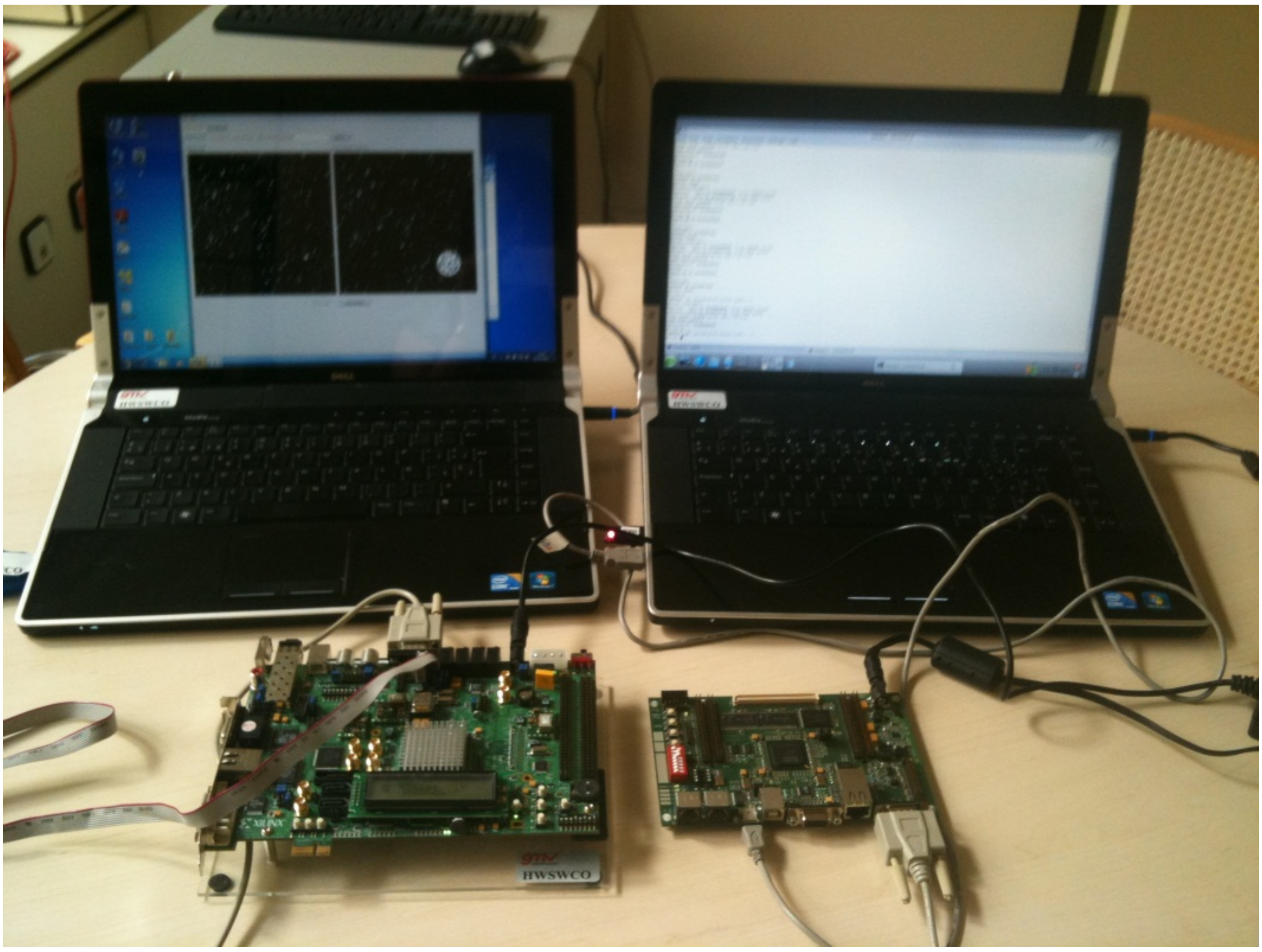


Figure 3: HWSWCO demonstrator: FPGA Virtex 5, Leon2 development board and PC acting as camera instrument.

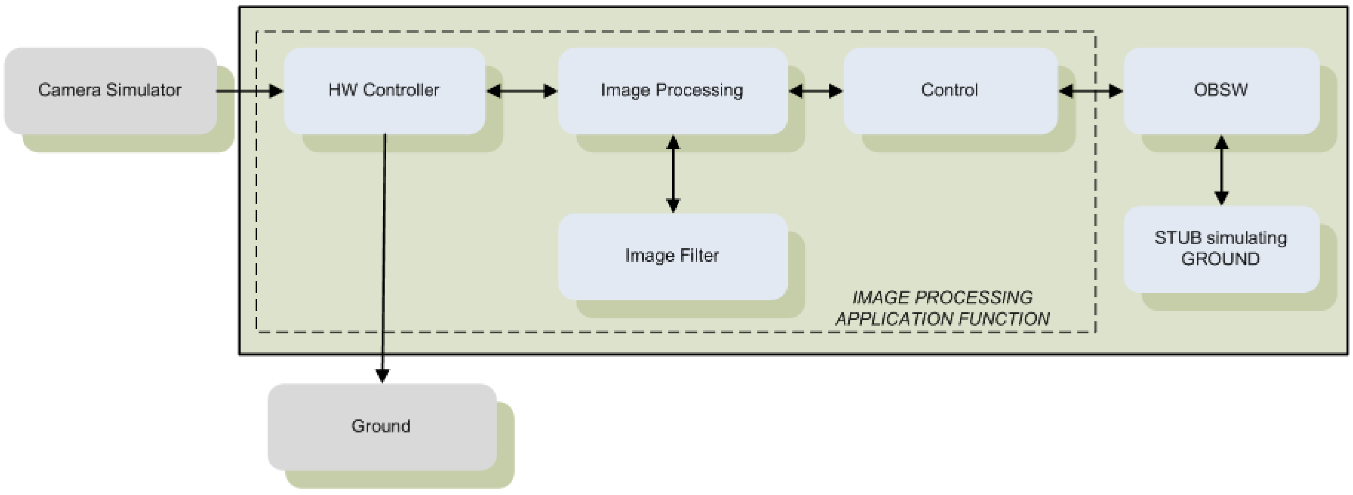


Figure 4: Image processing case-study.

- Image processing including image filtering.
- Interfacing with the On-Board SW (OBSW) for command and configuring messages.
- Interfacing with the OBSW for health status reporting.
- Configuring the image processing functions.
- Configuring the science packet reporting.

The system functions that are costly in terms of computation requirements are executed on an FGPA, while the OBSW is executed in a LEON2 processor. The OBSW will be responsible for the housekeeping, and the control and monitoring of the hardware system. The OBSW consists in two subcomponents (see figure 4):

Start-up. It is responsible for the start-up routine: simulates the initialization routine. It has only a subprogram called startexecution that is in charge of initializing the OBSW.

Image Processing Management. It commands the HW system to cyclically start/stop the image filtering functions and change the parameters of the filter. This is performed by sending the corresponding data packets to the HW system.

Moreover, the TASTE tool-chain generated the following set of files in order to define common types and parameters for the HW and SW systems:

asn1_types.ads It contains definition of common data types.

polyorb_hi_generated.ads Pure root of the generated PolyORB HI middleware hierarchy.

polyorb_hi_generated-deployment.ads It contains identifiers of the components and their interfaces deployed in the system. This information is used by the broker to communicate components and therefore, it is necessary to be known in the HW system.

PolyORB HI is a high integrity version of the PolyORB middleware [7] that is the part of the ASSERT Virtual Machine in charge of dealing with distribution. However, the current version of TASTE does not support communication through serial lines for embedded targets. The workaround was to develop the code semi-manually by using the definitions generated by TASTE and coding the activities following the Ravenscar computational model.

The component Image Processing Management was coded by following the archetype of figure 5. In order to provide communication by serial lines, an UART device driver [8] formerly developed for other LEON2 computer was adapted to the HWSWCO LEON2 board. Listing 1 shows the used archetype for the cyclic task that implements the Image Processing Management functionality.

The code of the Image Processing Management component is shown in listing 2. It must be noticed that a private child package called image_processing_management-packets.ads contains the definitions of the packets to be send to the HW system.

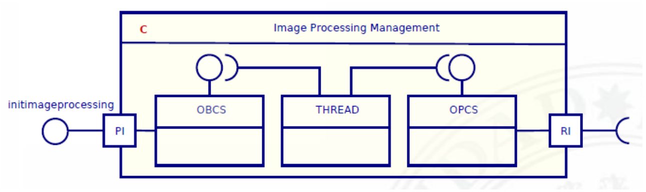


Figure 5: Image Processing Management component.

Listing 1: Component template for cyclic tasks

```
with System;
with Ada.Real_Time;

package Cyclic_Tasks is
   protected type Cyclic_OBCS
     ( Ceiling_Priority  :  System.Priority)
   is
      pragma Priority( Ceiling_Priority );
      procedure Signal;
      entry Wait;
   private
      Occurred : Boolean := False;
   end Cyclic_OBCS;

   task type Cyclic_Thread
     (Thread_Priority  :  System.Priority;
      Period              :  Positive;
                              -- milliseconds
      OBCS                :  access Cyclic_OBCS;
      Operation           :  access procedure)
   is
      pragma Priority(Thread_Priority);
   end Cyclic_Thread;

end Cyclic_Tasks;

package body Cyclic_Tasks is

   use Ada.Real_Time;
   Next_Time : Time;

   protected body Cyclic_OBCS is

      procedure Signal is
      begin
         Occurred := True;
         Next_Time := Ada.Real_Time.Clock;
      end Signal;

      entry Wait when Occurred is
      begin
         Occurred := False;
      end Wait;

   end Cyclic_OBCS;

   task body Cyclic_Thread is
   begin
      OBCS.Wait;
      loop
         delay until  Next_Time;
         Operation.all;
         Next_Time := Next_Time +
                          Milliseconds(Period);
      end loop;
   end Cyclic_Thread;

end Cyclic_Tasks;
```

Listing 2: Image Processing Management component

```
with Cyclic_Tasks; use Cyclic_Tasks;
package image_processing_management is

   OBCS : aliased Cyclic_OBCS
                  ( Ceiling_Priority => 2);

   procedure initimageprocessingi
            renames OBCS.Signal;

private

   procedure Periodic_Activity;

   Thread : Cyclic_Thread
     (Thread_Priority => 1,
      Period          => 40_000, -- ms
      OBCS            => OBCS'Access,
      Operation       => Periodic_Activity 'Access);

end image_processing_management;


with GlUart.HLInterface;
with image_processing_management.packets;
with System;

package body image_processing_management is

   package UHL renames GlUart.HLInterface;
   package PKG renames
           image_processing_management.packets;

   Uart_2 : UHL.Serial_Port;
   type Cycle_Type is mod 6;
   Current_Cycle : Cycle_Type := 0;

   procedure Periodic_Activity is
   begin
     case Current_Cycle is
       when 0 => UHL.Write (Uart_2,
                   PKG.Start_Packet);
                 UHL.Write (Uart_2,
                   PKG.Filter_0_Packet);
       when 1|4 => UHL.Write (Uart_2,
                   PKG.Filter_1_Packet);
       when 2 => UHL.Write (Uart_2,
                   PKG.Filter_2_Packet);
       when 3 => UHL.Write (Uart_2,
                   PKG.Filter_0_Packet);
       when 5 => UHL.Write (Uart_2,
                   PKG.Stop_Packet);
       end case;
     Current_Cycle := Current_Cycle + 1;
   end Periodic_Activity;

begin

   UHL.Open (Uart_2, UHL.UART_1);
   UHL.Set (Port => Uart_2,
           Rate => UHL.B115200);

end image_processing_management;
```

This case study exercises the whole methodology and rose important conclusions with regard to the interaction between the HW and SW synthesis phases.

5 Conclusions and future work

Four key points were identified in the development of a distributed HW/SW system compliant with the ASSERT process:

- The modeling environment should be able to generate HW data model compatible with ASSERT, especially in terms of data size and bit ordering.
- The HW system must be compliant with the ASSERT component model and have a similar programming model.
- The HW system must have a common representation of the system components so that the ASSERT HW broker is able to dispatch the services mapped to HW, and communicate with the SW system using the same communication protocol.
- Finally, it should use a similar programming language so that the effort of migrating system functions from SW to HW is minimized to the maximum extent.

However this study did not cover all aspects of the HW/SW co-development, and there are some missing issues to be covered in future lines of work. One of the most interested is the analysis and definition of a computational model for the hardware systems that were analyzable and compatible with the Ravenscar Computational Model of the SW system.

Acknowledgments

This work was supported by the HWSWCO project ("Hardware/Software Co-design"), ESA Technological Research Program (TRP) study, contract ESTEC 22810/09/JK, directed by Dc. Eric Conquet. The partners involved in this study were GMV as prime contractor, GMV Systems, the Universidad of Cantabria and the Universidad Politécnica of Madrid.

References

[1] M. Bordin and T. Vardanega (2007), *Correctness by construction for high-integrity real-time systems: A metamodel-driven approach*, in *12th International Conference on Reliable Software Technologies — Ada-Europe 2007* (N. Abdennadher and F. Kordon, eds.), no. 4498 in LNCS, pp. 114–127, Springer-Verlag, 2007.

[2] ISO, ISO/IEC TR 24718:2005 (2005) — *Guide for the use of the Ada Ravenscar Profile in high integrity systems*. Based on the University of York Technical Report YCS-2003-348 (2003).

[3] J. Zamorano, J. A. de la Puente, J. A. Pulido, and S. Urueña (2008), *The ASSERT virtual machine kernel: Support for preservation of temporal properties*' in *Data Systems in Aerospace — DASIA 2008*.

[4] ISO, ISO/IEC 8652:1995(E)/TC1(2000)/AMD1(2007): *Information Technology — Programming Languages — Ada*.

[5] SAE (2009), *SAE AS5506A Architecture Analysis and Design Language (AADL)*, January 2009. Available at `www.sae.org`.

[6] OMG (2003), *MDA Guide Version 1.0.1*. Available at `http://www.omg.org/mda/`.

[7] T. Vergnaud, J. Hugues, L. Pautet, and F. Kordon (2004), *PolyORB: a schizophrenic middleware to build versatile reliable distributed applications*, in *Proc. of the 9th International Conference on Reliable Software Techologies Ada-Europe 2004 (RST'04)*, vol. LNCS 3063, pp. 106–119, Springer Verlag.

[8] J. López, Á. Esquinas, J. Zamorano, and J. A. de la Puente (2010), *Experience in programming device drivers with the Ravenscar profile*, Ada User Journal, vol. 31.

Integrating 8-bit AVR Micro-Controllers in Ada

Pablo Vieira Rego
Embraer, S.A., Av. Brigadeiro Faria Lima 2170, Putim, São José dos Campos, SP, Brazil; email: pvrego@gmail.com

Abstract

The increasing popularity of 8-bit AVR micro-controllers for using with robotics and low-complexity embedded applications has called the attention of much of the software community. The purpose of this paper is to present an approach for developing embedded applications for these micro-controllers in Ada, using GNAT AVR as Microsoft Windows cross-compiler and GNAT Programming Studio as IDE.

Keywords: 8-bit AVR micro-controller, GNAT AVR, Ada, ZFP.

1 Introduction

The Atmel AVRs[1] are presented as a family of 8- and 32-bit micro-controllers designed to provide flexibility, based a priori in C and assembly programming. More specifically, Arduino is a family of boards with a central 8-bit AVR micro-controller and peripheral I/Os which implement a bunch of low-level hardware support such as analog and digital data write, read and storage, USART protocols, Ethernet, wifi modules and others, for a reasonable low cost. They became very popular in the development of memory mapped robotics with open-source hardware for hobbyists and low-to-medium complexity embedded products and introductory robotics programming courses [1, 2, 3, 4, 5].

For open-source development in C/C++, there are four main branches: Atmel Studio, WinAVR, Arduino IDE and Eclipse. Atmel Studio[2] provides the IDE with C and assembly compiler and debugger and several in-built libraries with pre-implemented features and specific configurations for the whole 8- and 32-bit families. WinAVR[3] is a suite of executable, open source software development tools for the Atmel AVR series of RISC microprocessors hosted on the Windows platform, which includes the GNU GCC compiler for C and C++. Arduino IDE[4] is a platform written in Java and based on Processing and avr-gcc and provides C++ development based on application examples. On Eclipse, the AVR Eclipse Plugin[5] provides tools for developing C applications on AVR micro-controllers. In all cases, Atmel's Application Notes serve as development guidelines, as well as the micro-controllers datasheets, which contain the complete hardware description and some code examples[6]. For RTOS'es support, there are plenty of choices for AVR development, we could cite AvrX[7], FreeRTOS[8] and RTEMS[9], most of them written in C/C++.

For the development in Ada, there are some academic references for development on AVRs, most focused on the 32-bit micro-controllers. Gregertsen and Skavhaug [6] describe a deterministic multitasking run-time environment supporting the Ravenscar tasking model of Ada 2005 implemented on the Atmel AVR32 UC3A micro-controller. In [7], they describe an object-oriented real-time framework for Ada 2005 and Ravenscar profile implemented on AVR32 UC3 micro-controller; in [8], they propose the addition of execution-time control features for interrupt handling as an addition to the Ada standard library and describes it using an implementation of GNAT bare-board Ravenscar run-time environment on the Atmel AVR32 architecture; and finally in [9, 10], Gregertsen and Skavgaug describe an implementation of timing event and execution time control features with support for interruption on the same architecture.

For the 8-bit micro-controllers, however, there are fewer academic references for the development in Ada. Ras and Cheng [11] describe a deterministic run-time environment for Ada-05 on the 8-bit ATmega16 micro-controller; and Andersen [12] presented an approach for developing Arduinos with the AVR-Ada cross-compiler.

There are some open-source cross-compiler options for programming AVR micro-controllers in Ada, we can cite AVR-Ada[10] and GNAT AVR GPL [13]. In this paper we will show an approach to generate a .hex target burning file for the micro-controller using the open-source GNAT AVR cross compiler and the GPS IDE. The GNAT AVR provides a Microsoft Windows cross-compiler and builder for AVR 8-bit micro-controllers [14] and implements *Zero Footprint Profile (ZFP)*, which does not require any run-time routines, and is intended for high-critically applications related to DEF Stan 00-55 certification [15]. Thus it does not have support for tasking or exception propagation. So, the purpose of this paper is to serve as an user guide for development of monotask applications in 8-bit AVR micro-controllers in Ada, following the ZFP.

[1] `http://www.atmel.com/products/microcontrollers/avr/default.aspx`
[2] `http://www.atmel.com/microsite/atmel_studio6/`
[3] `http://winavr.sourceforge.net/index.html`
[4] `http://arduino.cc/en/Main/Software`
[5] `http://avr-eclipse.sourceforge.net/wiki/index.php/The_AVR_Eclipse_Plugin`
[6] `http://www.atmel.com/products/microcontrollers/avr/default.aspx`
[7] `http://www.barello.net/avrx/index.htm`
[8] `http://www.freertos.org/`
[9] `http://www.rtems.com/`
[10] `http://sourceforge.net/apps/mediawiki/avr-ada/index.php?title=Main_Page`

2 Materials and Methods

2.1 Arduino Duemilanove (ATmega328P)

The Arduino Duemilanove is a development board based on Atmel ATmega328P/ATmega168 8-bit micro-controllers. This paper concerns to ATmega328P, but can be extended to other AVR 8-bit chips with minor changes. Duemilanove provides 14 digital I/O pins, 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header and a reset button[11]. Table 1 resumes some characteristics of this board.

Table 1: Characteristics of Arduino Duemilanove with ATmega328p

micro-controller	ATmega328P
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB[12]
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

2.2 Development Environment and Target Generation

The required applications are the GNAT AVR GPL edition[13], which comes with the IDE and the Windows cross-compiler; and WinAVR, which provides CRT libraries for the micro-controllers.

For generating the output .hex file on Windows, the simple project consists of the following elements:

- Specification package which will map the AVR registers;
- Main file which will include the chip specification package;
- GPS .gpr configuration file;
- An object library provided by WinAVR;
- A batch script which will provide .elf/.hex conversion and a pre-configured burning tool.

2.2.1 *The Specification Package*

Each AVR device has a number of I/Os associated with its internal registers. These descriptions are present in micro-controller datasheet and the specification package shall map the memory address entries defined in the datasheet.

For the ATmega328P micro-controller, datasheet [16] section *Register Summary* on page 423 describes all the memory addresses available to develop the applications. Using the Memory Mapped I/O approach as described by McCormick et al. [17], each word shall be mapped, and sometimes it is usefull to map also the meaning of each one of the bits. The words so should be pointed exactly to the memory address it describes.

For example, the registers *Port B Data Register (PortB)*, *Port B Data Direction Register (DDRB)* and *Port B Input Pins Address (PINB)* could be mapped in a specification package as

Listing 1: Simplified Specification Package for ATmega328P

```
-- atmega328p.ads
with Interfaces; use Interfaces;
with System;

package ATmega328P is

   -- PORTB: Port B Data Register
   PORTB : Unsigned_8;
   for PORTB'Address use System'To_Address (16#25#);

   -- DDRB: Port B Data Direction Register
   DDRB : Unsigned_8;
   for DDRB'Address use System'To_Address (16#24#);

   -- PINB: Port B Input Pins
   PINB : Unsigned_8;
   for PINB'Address use System'To_Address (16#23#);

end ATmega328P;
```

2.2.2 *GPS Project File*

The GPS project file has to include the AVR configurations for the packages *Ide*, *Compiler*, *Builder* and *Linker*. Below are presented some suggestions on how to configure these project file packages in order to build the .elf output which will be used to generate the .hex output.

Listing 2: GPS Project File for ATmega328P

```
-- arduino.gpr
project Arduino is

   for Source_Dirs use (".", "src");
   for Object_Dir use "obj";
   for Exec_Dir use "bin";
   for Main use ("main.adb");

   package Ide is
      for Gnat use "avr-gnat";
      for Gnatlist use "avr-gnatls";
      for Debugger_Command use "avr-gdb";
   end Ide;

   package Compiler is
```

[11]General description of Arduino Duemilanove can be found at http://arduino.cc/en/Main/arduinoBoardDuemilanove

[12]In which 2 KB is used by bootloader.

[13]http://libre.adacore.com

```
      for Default_Switches ("ada") use ("−mmcu=avr5");
   end Compiler;

   package Builder is
      for Executable_Suffix use ".elf";
      for Default_Switches ("ada") use ("−−RTS=rts−zfp");
   end Builder;

   package Linker is
      for Default_Switches ("ada") use ("obj\crtm328p._o", "
            −nostdlib", "−lgcc", "−mavr5", "−Tdata=0
            x00800200", "−−mmcu=avr5");
   end Linker;

end Arduino;
```

The *Ide* package configures the tools related to the AVR-GNAT installation. The *Builder* package sets the suffix for the builder output file and specifies the Zero Footprint Profile as the Real-Time System.

In the *Compiler* package, the option *-mmcu=avr5* enables compilation using AVR5 architecture, which is the case of the ATmega328P.

In the *Linker* package, the *crtm328p._o* is the CRT[14] got from WinAVR installation[15], since GNAT AVR GPL only provides CRT for ATmega2560. The option *-nostdlib* avoids linking with standard libraries, not provided by GNAT AVR GPL, and the option *-lgcc* tells the linker to use the compiler support library. The options *-mavr5* and *-mmcu=avr5* set the device architecture for AVR5. And the option *-Tdata* sets the start address of the data section [18].

For the ATmega2560 micro-controller, we should exchange from AVR5 to AVR6 architecture and the CRT. GNAT provides this CRT in a form of *crt1-atmega2560.S* file[16], which shall be compiled in order of obtaining the *crt2560._o*. One can compile it using

```
avr−gcc −c −mmcu=avr6 −o crt2560._o crtl−atmega2560.S
```

2.2.3 The Main File

The Main file contains the main task available by ZFP and the libraries related to the code. The following dummy code should be modified in order to code the micro-controller.

Listing 3: Dummy main file

```
with ATmega328P; use ATmega328P;

procedure Main is
begin
   null;
end Main;
```

[14]C Run-Time Library.

[15]In this case, the crtm328p.o has its extension renamed to _o to avoid been deleted when the user makes a project clean.

[16]The file *crt1-atmega2560.S* can be found on GNAT AVR GPL 2012 installation in the folder *C:\GNAT\2012\share\examples\gnat-cross\avr\atmega2560.*

2.2.4 The Hex & Burn Batch Script

An option for a batch script to generate the .hex and burn it on a micro-controller in COM7 follows.

Listing 4: Batch script for converting the .elf and burn the resulting .hex

```
@rem hexitandburn_duemilanove.bat

@echo Converts .elf to .hex
avr−objcopy −O ihex main.elf main.hex

@echo Burn the chip
c:\WinAVR\bin\avrdude.exe −p m328p −c avrisp −P com7
      −b 57600 −e −U flash:w:main.hex
```

In the case of using the ATmega2560 micro-controller, the *avrdude* line would change to

```
c:\WinAVR\bin\avrdude.exe −p m2560 −c stk500v2 −P
      com7 −b 115200 −e −U flash:w:bin\main.hex
```

And it is suggested to create a .xml file in Windows user profile folder (eg. c:\Documents and Settings\user\.gps\plug-ins) to enable a menu to execute the actions of convert and burn the .hex in the micro-controller.

Listing 5: Script embedded.xml to enable a menu in GPS to convert and burn Arduino Duemilanove

```
<?xml version="1.0" ?>
<embedded>
      <action name="Hexit␣and␣Burn␣Duemilanove">
            <external>cmd /c
                  hexitandburn_duemilanove.bat</
                  external>
      </action>

      <submenu after="Build">
            <title>Embedded</title>
            <menu action="Hexit␣and␣Burn␣
                  Duemilanove">
                  <title>[Hexit and Burn]
                        Duemilanove</title>
            </menu>
      </submenu>
</embedded>
```

3 Limitations of the Zero Footprint Profile

The Zero Footprint Profile defines an Ada run-time certifiable run-time with a memory footprint reduced to null. It excludes in particular the use of dynamic Ada semantic. The ZFP still allows the use of the major Ada features such as generics, child units, library-level tagged types, interfaces and local exception handling [19]. The ZFP is suitable for software certification using DEF Stan 00-55 standard [15].

The ZFP limitations are [20]:

- Constructs defined in the ARM Section 9 (Tasking and Synchronization)
- Exception propagation
- Packed arrays with component size other than 1, 2, 4, 8 or 16 bits

- Some exponentiation operation
- 64-bit integer and fixed-point types
- Boolean operation on packed arrays
- Some comparison operations on arrays of discrete components
- The attributes *Image*, *Value*, *Body_Version*, *Version*, *Width*, and *Mantissa*
- Controlled types
- Some applications of the attributes *Address*, *Access*, *Unchecked_Access*, *Unrestricted_Access*
- Non library-level tagged types
- Annex E (Distributed Systems)

4 An Example Implementation

In this section we will show a very simple example application. For this implementation, it is enough to use the *arduino.gpr* project file described in section 2.2.2 and the *atmega328p.ads* described in section 2.2.1 as specification package and a main file. So, suppose we desire to move a 5-wires Stepper Motor some steps forward, blink a led once, move the Stepper Motor some steps backward and blink the same led twice. The Stepper Motor used is a Mototech S35S5 extracted from an old optical scanner.

Figure 1 shows the connections among the components. The Arduino Duemilanove is connected to the Stepper Motor through the current driver ULN2803AP using pins B0, B1, B2 and B3, and the led is connected using pin B5.

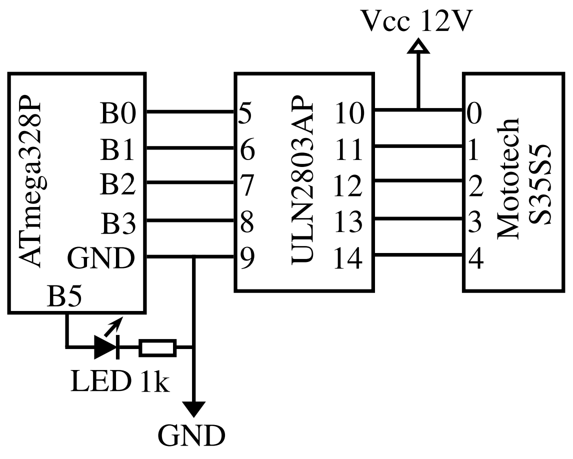


Figure 1: Connections between Arduino Duemilanove, Led, ULN2803AP and Stepper Motor.

```
-- main.adb
with ATmega328P; use ATmega328P;

procedure Main is

   -- Stepper Motor coils definitions
   COIL0 : constant := 2 ** 0;
   COIL1 : constant := 2 ** 2;
   COIL2 : constant := 2 ** 1;
   COIL3 : constant := 2 ** 3;

   -- Default delay and number of cycles
   DEFAULT_DELAY : constant := 100;
   DEFAULT_CYCLES : constant := 50;

   -- Null instruction delay
   procedure Custom_Delay (Wait_Cycle : Integer) is
   begin
      for i in 1 .. Wait_Cycle loop
         for j in 1 .. Wait_Cycle loop
            null;
         end loop;
      end loop;
   end Custom_Delay;

   procedure Blink_Times (Times : Integer) is
   begin
      for Counter in 1 .. Times loop
         PortB := 2#00100000#; Custom_Delay (5 *
               DEFAULT_DELAY);
         PortB := 2#00000000#; Custom_Delay (5 *
               DEFAULT_DELAY);
      end loop;
   end Blink_Times;

begin

   -- Initialize all PortB pins as outputs
   DDRB := 2#11111111#;

   -- Forward over the coils
   for Counter in 1 .. DEFAULT_CYCLES loop
      PortB := COIL0; Custom_Delay (DEFAULT_DELAY);
      PortB := COIL1; Custom_Delay (DEFAULT_DELAY);
      PortB := COIL2; Custom_Delay (DEFAULT_DELAY);
      PortB := COIL3; Custom_Delay (DEFAULT_DELAY);
   end loop;

   -- Blink led once
   Blink_Times (1);

   -- Backward over the coils
   for Counter in 1 .. DEFAULT_CYCLES loop
      PortB := COIL3; Custom_Delay (DEFAULT_DELAY);
      PortB := COIL2; Custom_Delay (DEFAULT_DELAY);
      PortB := COIL1; Custom_Delay (DEFAULT_DELAY);
      PortB := COIL0; Custom_Delay (DEFAULT_DELAY);
   end loop;

   -- Blink led twice
   Blink_Times (2);

   -- Disable all output pins
   PortB := 0;

end Main;
```

5 Conclusion

This paper shows an approach for developing Ada applications for 8-bit AVR micro-controllers, using GNAT AVR as the Microsoft Windows cross-compiler, and GNAT Programming Studio as IDE. GNAT AVR follows the Zero Footprint Profile for Real-Time Library, thus not all Ada Real-Time features are present in this development such as multi-tasks.

We believe that Ada can improve the quality of the applications for AVRs. The AVR community is invited to try this approach. Also the Ada community is encouraged to propagate its knowledge about real-time applications to the AVRs field in a mean that it certainly will gain more popularity over more one embedded software branch.

References

[1] J. D. Brock, R. F. Bruce, and S. L. Reiser (2009), *Using Arduino for introductory programming courses*, J. Comput. Sci. Coll., vol. 25, pp. 129–130.

[2] P. Jamieson, *Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?*. `http://www.users.muohio.edu/jamiespa/html_papers/fecs_11.pdf`. Accessed: 23/06/2012.

[3] R. Balogh, *Robotics course with the Acrob robot*. `http://www.innoc.at/fileadmin/user_upload/_temp_/RiE/Proceedings/15.pdf`. Accessed: 23/06/2012.

[4] R. Balogh (2010), *Acrob - an Educational Robotic Platform*, AT&P Journal Plus, vol. 10, no. 2, pp. 6–9. ISSN 1336-5010.

[5] I. B. Gartseev, L.F. Lee, and V. N. Krovi (2011), *A Low-Cost Real-Time Mobile Robot Platform (ArEduBot) to support Project-Based Learning in Robotics & Mechatronics*, pp. 117–124, INNOC - Austrian Society for Innovative Computer Sciences. Accessed: 23/06/2012.

[6] K. N. Gregertsen and A. Skavhaug (2009), *An efficient and deterministic multi-tasking run-time environment for Ada and the Ravenscar profile on the Atmel AVR32 UC3 microcontroller*, in *Proc. of Design, Automation Test in Europe Conference Exhibition*, pp. 1572 –1575.

[7] K. N. Gregertsen and A. Skavhaug (2009), *A Real-Time Framework for Ada 2005 and the Ravenscar Profile*, in *Proc. Software Engineering and Advanced Applications, 35th Euromicro Conference on*, pp. 515 –522.

[8] K. N. Gregertsen and A. Skavhaug (2010), *Execution-time control for interrupt handling*, Ada Lett., vol. 30, pp. 33–44.

[9] K. N. Gregertsen and A. Skavhaug (2010), *Implementing the new Ada 2005 timing event and execution time control features on the AVR32 architecture*, Journal of Systems Architecture, vol. 56, no. 10, pp. 509 – 522.

[10] K. N. Gregertsen (2012), *Execution Time Control : A hardware accelerated Ada implementation with novel support for interrupt handling*. PhD thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics.

[11] J. Ras and A. M. Cheng (2010), *A deterministic run-time environment for Ada-05 on the ATmega16 microcontroller*, Ada Lett., vol. 30, pp. 13–22.

[12] J. S. Andersen (2012), *Programming Arduinos in Ada*, Free and Open Source Software Developers' European Meeting (FOSDEM'12).

[13] AdaCore (2009), *GNATPro available for 8-bit AVR Microcontroller*. Accessed: 23/06/2012.

[14] AdaCore, *Ada Development Environment for the Atmel AVR 8-bit microcontroller*.

[15] Ministry of Defence (1991), *Requirements for Safety Related Software in Defense Equipment*.

[16] Atmel Corporation, *8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash: ATmega48PA/ ATmega88PA/ ATmega168PA/ ATmega328P*. Accessed: 21/06/2012.

[17] J. W. McCormick, F. Singhoff, and J. Hugues (2011), *Building Parallel, Embedded, and Real-Time Applications with Ada*. New York, USA: Cambridge University Press, first" ed..

[18] AdaCore, *AVR Topics: J.2 Compiler and Linker Flags for AVR*. `http://docs.adacore.com/gnat-cross-docs/html/gnat_ugx_12.html`. Accessed: 24/06/2012.

[19] AdaCore, *Runtime Profiles*. `http://www.adacore.com/gnatpro/toolsuite/runtimes/`. Accessed: 05/07/2012.

[20] AdaCore, *Ada Restrictions in the Zero Footprint Profile*. `http://docs.adacore.com/gnat-hie-docs/html/gnathie_ug_4.html#SEC27`. Accessed: 05/07/2012.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o KU 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. Hubert B. Keller
Karlsruher Institut für Technologie (KIT)
Institut für Angewandte Informatik (IAI)
Campus Nord, Gebäude 445, Raum 243
Postfach 3640
76021 Karlsruhe
Germany
Email: Hubert.Keller@kit.edu
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. Sergio Sáez
DISCA-ETSINF-Edificio 1G
Universitat Politècnica de València
Camino de Vera s/n
E46022 Valencia
Spain
Phone: +34-963-877-007, Ext. 75741
Email: ssaez@disca.upv.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: president@ada-switzerland.ch
URL: www.ada-switzerland.ch