

ADA USER JOURNAL

Volume 30
Number 4
December 2009

Contents

	<i>Page</i>
Editorial Policy for <i>Ada User Journal</i>	202
Editorial	203
News	205
Conference Calendar	230
Forthcoming Events	236
T. Vardanega <i>“Book review: Ada for Software Engineers, by Mordechai Ben-Ari”</i>	241
Articles	
M. Sobczak <i>“Experiences in Evaluating Ada with a Pilot Project”</i>	243
Articles from the Industrial Track of Ada-Europe 2009	
M. Bordin, C. Comar, T. Gingold, J. Guitton, O. Hainque, T. Quinot, J. Delange, J. Hugues, L. Pautet <i>“Couverture: an Innovative Open Framework for Coverage Analysis of Safety Critical Applications”</i>	248
T. Vergnaud, F. Gilliers, H. Balp <i>“Generating Component-based AADL Applications with MyCCM-HI and Ocarina”</i>	256
M. Aldea Rivas, M. González-Harbour <i>Ada User Guide on MaRTE OS</i>	264
Ada-Europe Associate Members (National Ada Organizations)	272
Ada-Europe 2009 Sponsors	Inside Back Cover

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

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

News and Product Announcements

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

Reprinted Articles

While original material is our first priority, we are willing to reprint (with the permission of the copyright holder) material previously submitted elsewhere if it is appropriate to give it a wider audience. This includes papers published in North America that are not easily available in Europe.

We have a reciprocal approach in granting permission for other publications to reprint papers originally published in *Ada User Journal*.

Commentaries

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

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

Announcements and Reports

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

Reviews

Inclusion of any review in the Journal is at the discretion of the Editor.

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

Submission Guidelines

All material for publication should be sent to the Editor, preferably in electronic format. The Editor will only accept typed manuscripts by prior arrangement.

Prospective authors are encouraged to contact the Editor by email to determine the best format for submission. Contact details can be found near the front of each edition.

Example papers conforming to formatting requirements as well as some word processor templates are available from the editor. There is no limitation on the length of papers, though a paper longer than 10,000 words would be regarded as exceptional.

Editorial

Browsing the contents of Volume 30 of the Ada User Journal, which closes with this issue, the reader will surely understand if I note that the Journal is demonstrating its vitality and, more important, the vitality of the Ada community.

In this Volume of four issues and 272 pages, fifteen articles were published; seven derived from the Ada-Europe conference Industrial and Tutorial tracks, five in the Proceedings of the Software Vulnerabilities Workshop, and three directly provided to the Journal. At the same time the Journal published eight Ada Gems, and innumerable news and event information. And we end the year inaugurating a new section: the *Ada User Guides*, which will provide to our readers hands-on guides of interesting Ada technologies, sharing the Journal with the Ada Gems section.

It is my pleasure to say that the Journal is very much alive, as is the Ada community.

As for this last issue of 2009, the first paper is by Maciej Sobczak, from CERN, Switzerland, and presents the author's experience in using the Ada language for a distributed directory and name service. We also finalize the publication of material derived from the Ada-Europe 2009 conference, with two papers from the conference's industrial track. The first, from a set of authors coming from AdaCore and Télécom ParisTech, both in France, presents a framework for coverage analysis of high-integrity applications. The second, from authors coming from Thales Communications, France, provides the results of combining Lightweight CCM and AADL for the development of real-time applications.

In the new *Ada User Guides* section, we are happy to provide the Ada User Guide on MaRTE OS, by Mario Aldea and Michael González-Harbour, of the University of Cantabria, Spain. I am sure that you will all appreciate, and use, this guide. Concerning the Guides section, Please feel free to write to us presenting or proposing other guides for us to publish.

Also in the issue, Tullio Vardanega provides an interesting Book Review of the second edition of Mordechai Ben-Ari's *Ada for Software Engineers*. And finally, I would like also to point out the, as usual, richness of the news, calendar and forthcoming events sections.

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

News

Marco Panunzio

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

Contents

Ada-related Organizations	205
Ada-related Events	206
Ada-related Tools	207
Ada-related Products	209
Ada and GNU/Linux	212
Ada and Microsoft	212
References to Publications	214
Ada Inside	214
Ada in Context	216

Ada-related Organizations

Ada-France — Status of the Association

From: Jérôme Hugues <hugues@telecom-paristech.fr>

Date: Mon, 7 Sep 2009 16:24:00 CEST

Subject: [ada-france] Assemblée extraordinaire le 21/09 à 18h30, à l'ENST

Mailing list: ada-france.org

[...]

As you know, this summer we launched a call for volunteers to continue the Association.

We also convened an Assembly on August 31 2009, to discuss the future of the Association.

That meeting effectively took place, but with very low attendance. Besides the members of the Board of Directors (Fabrice Kordon, Laurent Pautet, Frank Singhoff and myself), only one member (Jean-Pierre Rosen) was present. We reminded of the reasons why the current team (Yvon Kermarrec, Frank and myself) does not wish to continue anymore.

Formally, there is a lack of time to properly manage the Association.

As I indicated at the meeting, the Association can continue to provide certain basic services. However this can not happen without a team which will take over.

That is why we are convening a new Assembly with the dissolution of the Association as the single item on the agenda.

If a team would be formed by then to take over some of the activities of the Association, it is obviously welcome.

From: Jean-Pierre Rosen

<rosen@adalog.fr>

Date: Wed, 23 Sep 2009 17:55:00 CEST

Subject: [ada-france] Résultat de l'AG

Mailing list: ada-france.org

With a unanimous vote of present and represented members, the General Assembly rejected the proposal to disband the Association.

A new Board of Directors was elected, composed of:

- Thomas De Contes

- Xavier Grave

- Laurent Guerby

- Jean-Pierre Rosen

- Samuel Tardieu

The Board of Directors will now get organized to form the Executive Board, prepare the official paperwork, and resume activities. Of course, we will keep the list informed, but it will nevertheless take some time.

Stay tuned!

From: Jérôme Hugues <hugues@telecom-paristech.fr>

Date: Mon, 5 Oct 2009 9:11:00 CEST

Subject: [ada-france] CR de l'Assemblée Générale d'AdaFrance du 21/09/2009

Mailing list: ada-france.org

[...]

Enclosed, you find the minutes of the General Assembly held on 21/09/2009. Good reading and good luck to the new Board.

Minutes of the General Assembly of Ada-France, convened on September 21, 2009 at the École Nationale Supérieure des Télécommunications.

Present:

- Jérôme Hugues

- Fabrice Kordon

- Jean-Pierre Rosen

- Claude Simon, non member

- Samuel Tardieu

Represented with a valid proxy, including by e-mail:

- Michel Gauthier

- Laurent Guerby

- Claude Kaiser

- Patrick de Bondeli

- Philippe Tarroux

- Guillaume Foliard

as well as Agusti Canals, who communicated via mail his wish to see the Association continue.

Total: 6 members present or represented.

The President, noting that the General Assembly was properly convened in conformance with Article 14 of the statutes, opened the meeting at 18:30.

The secretary for the meeting is Jérôme Hugues.

The single item on the agenda is the dissolution of the Association.

The current team has recalled the past activities (the Ada-France day, the organization of RST'2009 [sic] in Brest), and the follow-up of daily business. The lack of involvement in the Association has motivated the current team to want to hand over.

Jean-Pierre Rosen pointed out to have gathered a team, consisting of:

- Thomas De Contes

- Xavier Grave

- Laurent Guerby

- Jean-Pierre Rosen

- Samuel Tardieu

This newly formed team will focus more on the promotion of the Ada language in France.

This new team being presented, the members present or represented have unanimously rejected the proposal to dissolve the Association, and have unanimously voted in favor of this new team.

There being no other items on the agenda, the meeting was closed at 19:15.

Good luck to the new Board.

[The whole thread was translated from French —mp]

Ada in Denmark

From: Thomas Løcke <tl@ada-dk.org>

Date: Mon, 07 Sep 2009 13:52:35 +0200

Subject: Ada in Denmark and a Wiki article

Newsgroups: comp.lang.ada

[...]

For the past few years, Ada in Denmark has been without a website. This is no longer the case, as <http://ada-dk.org> relaunched a few months ago.

Hopefully we will be able to attract more danish Ada programmers, now that we're

online again.

If you're a danish Ada programmer, please consider joining Ada in Denmark: <http://ada-dk.org/?page=join>

We'd love to have more members.

The new website comes with a Wiki. It is our goal to add beginner-friendly articles to this Wiki, as time permits. The first article is online and can be found at

<http://wiki.ada-dk.org/index.php/Ada.Containers.Vectors>

In case it isn't obvious from the above URL, this article is about the Ada.Containers.Vectors package. The goal of the article is to *show*, with actual code, beginners how the various procedures/functions in the Vectors package work.

If there are any glaring mistakes in the article, please let me know, and I'll fix them ASAP. Or you can just sign up, and add/remove/alter the page as you wish. It is a Wiki after all. :o)

Please bear in mind, that the article is still being edited by someone with a much higher degree of proficiency in the English language (thanks Dwight!) than me. My first language, naturally, is Danish, so there's bound to be quite a few errors in the text, as I struggle along with the English language. Hopefully Dwight will have most of the errors cleared away "soon".

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 7 Sep 2009 06:53:20 -0700 PDT
Subject: Re: Ada in Denmark and a Wiki article
Newsgroups: comp.lang.ada

> The new website comes with a Wiki. It is our goal to add beginner-friendly articles to this Wiki, as time permits. The first article is online and can be found at <http://wiki.ada-dk.org/index.php/Ada.Containers.Vectors>

Nice but I think it would be better if the Ada in Denmark web site would concentrate on subjects that are specifically Danish (and quite possibly written in Danish, too; there is nothing wrong with that); your nice page about Vectors is relevant to users of Ada outside Denmark and is written in English, so I think this page should be in the Ada Programming wikibook.

Anyway, you have my sympathy if that matters to you :)

From: Thomas Locke <tl@ada-dk.org>
Date: Mon, 07 Sep 2009 16:27:41 +0200
Subject: Re: Ada in Denmark and a Wiki article
Newsgroups: comp.lang.ada

[...]

We talked about doing the website in danish, but decided on English, as all danish programmers are fairly proficient

in English, and if the site is in English, other people can benefit from the content also. It's a win/win situation. Had we done the website in danish, only very few people would benefit from the content.

> your nice page about Vectors is relevant to users of Ada outside Denmark and is written in English, so I think this page should be in the Ada Programming wikibook.

When we planned the new website, we did discuss whether a new wiki was necessary or if we should just link directly to, for example, the Ada Programming wikibook, but we decided on going with our own, as we wanted freedom to do whatever we liked, and we guessed (maybe wrongly) that such freedom would not be possible, if we were to intrude on an already active wiki.

With our own wiki, we can write whatever we want, however we want. And if other people feel some of our content is good enough to be added to for example the Ada Programming wikibook, then they are more than welcome to grab the content in question and add it. Copy it, modify it, use it. All content on the ada-dk.org wiki is made available under the GFDL 1.3 license.

[...]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 7 Sep 2009 08:49:48 -0700 PDT
Subject: Re: Ada in Denmark and a Wiki article
Newsgroups: comp.lang.ada

[...]

Yes, you guessed wrongly. Everyone is welcome to contribute to the Ada Programming wikibook. The existing page about Vectors[1] is very short and would benefit greatly from your work. Meanwhile, your page is probably not the first place newbies would find when looking for information whereas, even a few minutes ago, someone posted a question right here on comp.lang.ada after reading the now well-known Wikibook.

[1] http://en.wikibooks.org/wiki/Ada_Programming/Libraries/Ada.Containers.Vectors

> if we were to intrude on an already active wiki.

You have two people's sympathy already. You're not an intruder. And the Ada Programming wiki is never active enough. :)

> With our own wiki, we can write whatever we want, however we want. And if other people feel some of our content is good enough to be added to for example the Ada Programming wikibook, then they are more than welcome to grab the content in question and add it. Copy it, modify it, use it. All

content on the ada-dk.org wiki is made available under the GFDL 1.3 license.

That's called "duplication of effort" and "maintenance nightmare". Ada was designed to maximize reuse and minimize the maintenance burden. Heed her advice :)

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Mon, 07 Sep 2009 18:05:55 +0200
Subject: Re: Ada in Denmark and a Wiki article
Newsgroups: comp.lang.ada

[...]

Duplication of efforts is a waste, but duplication of web sites makes Ada more visible...

Ada-related Events

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

Ada at FOSDEM 2010 — Call for Interest

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Mon, 2 Nov 2009 05:09:25 +0100 CEST
Subject: Ada at FOSDEM 2010 - Call for Interest
Newsgroups: comp.lang.ada, fr.comp.lang.ada

Call for Interest

A d a at F O S D E M 2 0 1 0

6-7 February 2010, Brussels, Belgium

 FOSDEM [1], the Free and Open source Software Developers' European Meeting, is a free and non-commercial two-day event organized each February in Brussels, Belgium.

The goal is to provide Free Software and Open Source developers and communities a place to meet with other developers and projects, to be informed about the latest developments in the Free Software and Open Source world, to attend interesting talks and presentations by Free Software and Open Source project leaders and committers on various topics, and to promote the development and the benefits of Free Software and Open Source solutions.

At previous FOSDEM events, Ada-Belgium [2] has organized some very well attended Ada Developer Rooms, offering a full day program in 2006 [3] and a two-day program earlier this year [4].

Each year the number of applications for DevRooms outnumbers the available space, presenting the organizers with a difficult selection[5]. For 2010, the conditions specify "a preference for requests with a general topic, e.g. from projects with similar goals/domains" and "be involved in Free or Open Source Software (the projects produce and release software under an open source license or otherwise contribute to open source activities and communities)". Many Ada-related topics and projects fit those conditions very well, so we are considering to submit a proposal for FOSDEM 2010, and thus need to show that this would attract sufficient interest.

To increase our chances to be allocated a DevRoom, we'd like to have a proposal with the full schedule of all presentations ready by the deadline for DevRoom requests.

Therefore, Ada-Belgium calls on you to:

- Inform us at ada-belgium-board@cs.kuleuven.be about specific presentations you would like to hear in an Ada DevRoom.
- For bonus points, subscribe to the Ada-FOSDEM mailing list [6] to discuss and help organize the details.
- For more bonus points, be a speaker: the Ada-FOSDEM mailing list is the place to be!

We look forward to lots of feedback! Please act ASAP and definitely before November 9.

The FOSDEM Team of Ada-Belgium
PS: This Call for Interest is also available online [7], including versions in PDF format suitable for printing (72 KB) and in plain text format for further distribution (4 KB).

- [1] <http://www.fosdem.org>
- [2] <http://www.cs.kuleuven.be/~dirk/ada-belgium>
- [3] <http://www.cs.kuleuven.be/~dirk/ada-belgium/events/06/060226-fosdem.html>
- [4] <http://www.cs.kuleuven.be/~dirk/ada-belgium/events/09/090207-fosdem.html>
- [5] <http://www.fosdem.org/2010/call-developer-rooms>
- [6] <http://listserv.cc.kuleuven.be/archives/adafosdem.html>
- [7] <http://www.cs.kuleuven.be/~dirk/ada-belgium/events/10/100206-fosdem.html>
- [...]

Review of Ada Issues

*From: Dirk Craeynest
<Dirk.Craeynest@cs.kuleuven.be>
Date: Mon, 19 Oct 2009, 22:22 GMT
Subject: Review of Ada Issues for November 2009 SC22/WG9 meeting (fwd)
Mailing list: ada-belgium-info@cs.kuleuven.ac.be*

Dear Ada-Belgium friend,

The following message was just posted to the Ada-Belgium members mailing list and is reposted here for your information.

[...]

Dear Ada-Belgium member,

As you may know, there is an upcoming meeting of ISO's Ada language working group (ISO/IEC JTC1/SC22/WG9) scheduled at the end of the ACM SIGAda 2009 conference next June in Tampa, Florida, USA

The Chairman of the Ada Rapporteur Group (ARG) of WG9 informed the Heads of Delegation that the Ada Issues (AIs) listed below have entered Editorial Review, and are intended to be submitted to WG9 for approval at the above mentioned meeting.

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

- AI05-0001-1/07 2009-07-03 - Bounded containers and other container issues
- AI05-0102-1/03 2009-06-27 - Some implicit conversions ought to be illegal
- AI05-0107-1/03 2009-06-27 - A failed allocator need not leak memory
- AI05-0123-1/06 2009-06-29 - Composability of equality
- AI05-0130-1/03 2009-06-26 - Order of initialization/finalization of record extension components
- AI05-0137-1/03 2009-06-30 - String encoding package
- AI05-0148-1/05 2009-06-25 - Accessibility of anonymous access stand-alone objects
- AI05-0152-1/02 2009-06-25 - Restriction No_Anonymous_Allocators
- AI05-0156-1/02 2009-06-25 - Elaborate_All applies to bodies imported with limited with

Those AIs are now being circulated within the Ada community for review, with the intention to return comments to the ARG in time to properly answer them before the WG9 meeting.

Comments for the Belgian delegation should be sent to me at Dirk.Craeynest@cs.kuleuven.be. The deadline is 18:00 GMT+2, Tuesday, October 27th, 2009. Early comments are encouraged.

Dirk Craeynest

ISO/IEC JTC1/SC22/WG9, Head of Delegation, Belgium

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/-Europe/SIGAda/WG9 mail)

[...]

[see also "Review of Ada Issues" in AUJ 30-2 (Jun 2009), p.72 —mp]

Ada-related Tools

GtkAda Contributions v2.5

*From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Wed, 18 Nov 2009 22:21:37 +0100
Subject: ANN: GtkAda contributions v2.5
Newsgroups: comp.lang.ada*

This library is proposed as a contribution to GtkAda, an Ada bindings to GTK+. It deals with the following issues:

1. Tasking support;
2. Custom models for tree view widget;
3. Custom cell renderers for tree view widget;
4. Multi-column derived model;
5. Extension derived model (to add columns to an existing model);
6. Abstract caching model for directory-like data;
7. Tree view and list view widgets for navigational browsing of abstract caching models;
8. File system navigation widgets with wildcard filtering;
9. Resource styles;
10. Capturing resources of a widget;
11. Embeddable images;
12. Some missing subprograms and bug fixes;
13. Measurement unit selection widget and dialogs;
14. Improved hue-luminance-saturation color model;
15. Simplified image buttons and buttons customizable by style properties;
16. Controlled Ada types for GTK+ strong and weak references;
17. Simplified means to create lists of strings;
18. Spawning processes synchronously and asynchronously with pipes;
19. Capturing asynchronous process standard I/O by Ada tasks and by text buffers;
20. Source view widget support.

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

Focus of this release is advanced debugging support including source navigation from tracing dialog (via communication with GPS)

[see also "GTKAda Contributions v2.4" in AUJ 30-3 (Sep 2009), p.144 —mp]

On the status of GNADE in Debian

From: Stephen Leake

*<stephen_leake@stephe-leake.org>
Date: Tue, 08 Sep 2009 07:18:36 -0400
Subject: users of GNADE MySQL binding?
Newsgroups: comp.lang.ada*

I'm considering becoming the Debian maintainer of GNADE.

In going over the code while preparing a release, I realized that the C structs that had been duplicated in Ada records in the GNADE MySQL binding are now wrong; they have changed significantly with each MySQL release, and GNADE has not kept up.

Therefore the MySQL binding in GNADE 1.5.3 (currently in Debian Lenny) most likely doesn't work, so I'm thinking no one uses it.

Rather than fixing it now, I'm considering dropping that library from the Debian release. I'll still package the ODBC binding, which does work with MySQL.

Any objections? Comments?

I still need to review the other GNADE bindings, to Sqlite3 and Postgre, for similar issues.

Previous versions of the GNADE Debian package included the Sqlite binding; this will be the first to include Sqlite3 instead. Does anyone still use Sqlite (not Sqlite3)? It would not be very hard to include both, aside from the effort required to review the C bindings.

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>

*Date: Sun, 13 Sep 2009 01:55:07 +0200
Subject: Re: users of GNADE MySQL binding?*

Newsgroups: comp.lang.ada

[...]

> Any objections? Comments?

No objection from me, as I'm only using the ODBC binding.

Have you seen my patch to the size comparisons in the generic versions of SQLBindCol and SQLBindParameter?

From: Stephen Leake

*<stephen_leake@stephe-leake.org>
Date: Sun, 13 Sep 2009 06:54:51 -0400
Subject: Re: users of GNADE MySQL binding?*

Newsgroups: comp.lang.ada

[...]

Yes, it's in my current sources.

OpenToken 3.1a

From: Stephen Leake

*<stephen_leake@stephe-leake.org>
Date: Tue, 08 Sep 2009 07:04:20 -0400*

*Subject: opentoken release
Newsgroups: comp.lang.ada*

I've finished the opentoken release. 3.1a is available at:

<http://www.stephe-leake.org/ada/opentoken.html>

Reto Buerki is packaging it for Debian.

I have quite a bit of work done towards the next release (probably numbered 4.0).

[see also "OpenToken" in AUJ 29-4 (Dec 2008), p.232 —mp]

Configuration file manager

From: Gautier de Montmollin

*<gdemont@users.sourceforge.net>
Date: Tue, 8 Sep 2009 03:01:22 -0700 PDT
Subject: Ann: Configuration file manager, v.02*

Newsgroups: comp.lang.ada

Hi - this very useful lightweight package from Rolf Ebert is on SourceForge for a while but was not advertised here yet.

URL:

<http://sourceforge.net/projects/ini-files/>

Config is a package for parsing configuration files (.ini, .inf, .cfg, ...) and retrieving keys of various types. New values for single keys can be set.

Standalone and unconditionally portable code.

Ahven 1.7

From: Tero Koskinen

*<tero.koskinen@iki.fi>
Date: Fri, 18 Sep 2009 22:01:04 +0300
Subject: ANN: Ahven 1.7
Newsgroups: comp.lang.ada*

[...]

I am pleased to announce Ahven 1.7. This is mostly a bug fix release.

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

The changes include a fix for Constraint_Error with long test names and special character filtering from the test names when generating XML results. In addition, PDF report generation example was added to the contrib directory and some internal code cleanups were done.

The source code can be downloaded from:

<http://sourceforge.net/projects/ahven/files/>

[see also "Ahven — Unit Test Library" in AUJ 30-1 (Mar 2009), p.11 —mp]

Zip-Ada

From: Gautier de Montmollin

*<gdemont@hotmail.com>
Date: Mon, 2 Nov 2009 13:17:51 -0800 PST
Subject: Ann: Zip-Ada v.33
Newsgroups: comp.lang.ada*

[...]

A new version of the Zip-Ada library, @ <http://unzip-ada.sf.net/>, is out. Latest changes:

- major performance improvement: decompression ~10x faster, compression ~3x faster (GNAT)
- ReZip: HTML display improved

*From: Gautier de Montmollin
<gdemont@hotmail.com>*

*Date: Sat, 21 Nov 2009 05:48:16 -0800 PST
Subject: Ann: Zip-Ada v.36
Newsgroups: comp.lang.ada*

Again a new version of the Zip-Ada library, @ <http://unzip-ada.sf.net/>, is out. Latest changes:

- BZip2 method added for decompression
- Added Zip.Traverse_verbose
- Added an UnZip.Extract to extract all files, using a Zip_Info variable (allows any stream, not only file, for archive)
- Some more run-time library performance bottlenecks removed (less spectacular than for v.35)
- Some improvements around ReZip, which now includes the BZip2 method

Zip-Ada is a library for handling, decompressing and creating .zip archives.

Some features:

- decompression for all methods up to BZip2 (new!)
- full sources are in Ada
- unconditionally portable
- input and output can be any stream (file, buffer,...) for archive creation as well as data extraction.
- task safe
- endian-neutral

URL: <http://unzip-ada.sf.net/>

The zipada36.zip archive contains:

- The full library sources inside one directory, Zip_Lib, in pure Ada 95
- Some command-line demo / tools:
 - o ZipAda, a zipping tool - only weak compression available so far
 - o UnZipAda, an unzipping utility
 - o Comp_Zip, compares two zip files (compare contents, check missing files)
 - o Find_Zip, searches a text string through contents of a zip file
 - o ReZip.adb, recompresses Zip archives

[see also "Zip-Ada v.33" in AUJ 30-3 (Sep 2009), p.147 —mp]

Strings Edit for Ada v2.3

From: Dmitry A. Kazakov

*<mailbox@dmitry-kazakov.de>
Date: Sun, 15 Nov 2009 14:20:33 +0100
Subject: ANN: Strings Edit for Ada v 2.3
Newsgroups: comp.lang.ada*

Provides string editing:

1. Integer numbers (generic, package Integer_Edit);
2. Integer sub- and superscript numbers;
3. Floating-point numbers (generic, package Float_Edit);
4. Roman numbers (the type Roman);
5. Strings;
6. Ada-style quoted strings;
7. UTF-8 encoded strings;
8. Unicode maps and sets;
9. Wildcard pattern matching.

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

Changes to the previous version:

1. An implementation of string streams was added.

From: John B. Matthews
<jmatthews@wright.edu>

Date: Sun, 15 Nov 2009 14:01:47 -0500

Subject: Re: ANN: Strings Edit for Ada v 2.3
Newsgroups: comp.lang.ada

[...]

Thank you for this instructive addition to your library and your generous license terms. The tests ran correctly using GNAT 4.3.4 (FSF) on Mac OS X 10.5.8.

Permit me to ask two questions: Would it be useful to add a line to the file test_strings_edit/readme_strings_edit.txt for the stream test? For example,

```
gnatmake -I./ test_string_streams.adb
```

Regarding the cautionary note in section 10 of the documentation, would it be correct to interpret the warning as "this implementation requires that Stream_Element'Size be a multiple of Character'Size and that the latter be a multiple of Storage_Element'Size."

[...]

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>

Date: Sun, 15 Nov 2009 21:10:35 +0100

Subject: Re: ANN: Strings Edit for Ada v 2.3
Newsgroups: comp.lang.ada

[...]

> Thank you for this instructive addition to your library and your generous license terms. The tests ran correctly using GNAT 4.3.4 (FSF) on Mac OS X 10.5.8.

Permit me to ask two questions: Would it be useful to add a line to the file test_strings_edit/readme_strings_edit.txt for the stream test? For example,

```
gnatmake -I./ test_string_streams.adb
```

Yes, I have changed the file. (I forgot to add instructions for gnatmake. I became lazy using GPS... (-))

> Regarding the cautionary note in section 10 of the documentation, would it be correct to interpret the warning as "this

implementation requires that Stream_Element'Size be a multiple of Character'Size and that the latter be a multiple of Storage_Element'Size."

I have changed the wording.

Thank you for the feedback. The changes will appear in the next version.

Ada-related Products

AdaCore — New Release of GNATbench for Wind River Workbench

From: AdaCore Press Center

Date: Fri, 18 Sep 2009

Subject: New Release of GNATbench for Wind River Workbench

URL: <http://www.adacore.com/2009/09/18/gnatbench-2-3-1-2/>

NEW YORK, PARIS and FRAMINGHAM, Mass., September 17, 2009 - Wind River Aerospace and Defense Regional Conference - AdaCore, a leading supplier of Ada development tools and support services, today announced a new release of its Ada Integrated Development Environment (IDE) plug-in, GNATbench 2.3.1.

This new release supports Workbench 3.1 and VxWorks 6.7, the latest versions of Wind River's Eclipse-based IDE and real-time operating system, offering real-time/embedded systems developers a sophisticated Ada programming environment tightly integrated into the Wind River Workbench development suite.

It also supports Workbench 3.0 and VxWorks 6.6, allowing projects that have chosen those versions to upgrade to GNATbench 2.3.1 and take advantage of its new features.

The GNATbench plug-in provides editing, browsing, and building features for Ada development (including Ada 2005) using AdaCore's GNAT Pro toolset on the Eclipse platform. The builder produces executables for native systems and embedded processors (in the context of Wind River Workbench), and likewise the debugger supports both native and embedded system debugging.

GNATbench 2.3.1 has introduced a variety of enhancements that help Ada software development run more smoothly. Creating new Workbench projects for Ada is now much simpler and more robust; there is no need to duplicate the use of new-project wizards in both Workbench and GNATbench. Error handling in the Import Wizard is friendlier, since GNATbench parses an imported project file for errors before attempting the import. The sharing of projects among multiple developers using distinct Workbench (Eclipse) workspaces

is simplified, since the values of a project's scenario variables (when changed from their defaults) are now stored in workspace-persistent variables. And as an open source project, the code of the implementation has been reorganized so that users wishing to extend the implementation can clearly identify which parts may be relied upon to remain stable in the future.

"AdaCore is pleased to be able to support the latest versions of Workbench and VxWorks in our new release of GNATbench," said Dr. Patrick Rogers, GNATbench Project Lead. "Ada is a key language for developers of real-time embedded systems using Workbench on Wind River platforms, and GNATbench provides Ada programmers with an intuitive and productive extension to Workbench. We look forward to continuing GNATbench enhancements in the future."

"GNATbench's support for our latest version of Workbench will be a great benefit to our Ada customers," said Mr. Rob Hoffman, Vice President and General Manager of Aerospace and Defense at Wind River. "We like our corporate partners to stay in sync with our product releases, and we appreciate AdaCore's consistent history of supporting new Wind River products and version releases as soon as they become available."

About AdaCore

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

Ada and GNAT Pro see a growing usage in high-integrity and safety-certified applications, including commercial aircraft avionics, military systems, air traffic management/control, railroad systems, and medical devices, and in security-sensitive domains such as financial services. The SPARK Pro toolset, available from AdaCore, is especially useful in such contexts.

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

www.adacore.com

AdaCore — GNAT Pro for LynxOS 5.0

From: AdaCore Press Center

Date: Tue, 22 Sep 2009

Subject: AdaCore Announces Release of GNAT Pro for LynxOS 5.0
URL: <http://www.adacore.com/2009/09/22/adacore-announces-release-of-gnat-pro-for-lynxos-50/>

NEW YORK, PARIS and BOSTON,
 September 22, 2009 - Embedded Systems Conference -

AdaCore, a leading supplier of Ada development tools and support services, today announced the release of the GNAT Pro Ada development environment for the LynxWorks LynxOS 5.0 operating system. This release allows GNAT Pro users to develop applications for both LynxOS 4.x and 5.0, and also provides a smooth migration path from older versions of the operating system to LynxOS 5.0. GNAT Pro for LynxOS 5.0 is available for both Linux and Windows host platforms, and for both PowerPC and x86 embedded targets.

The new release is completely compatible with the existing GNAT Pro 6.2.2 for LynxOS 4.x. GNAT Pro customers using this earlier LynxOS version can now take advantage of the many new features offered by LynxOS 5.0, including:

- Increased RAM support—up to 2 GB
- Symmetric Multiprocessing (SMP)
- ELF file format
- New POSIX—POSIX 1003.1-2003 PSE 53/54

“LynxOS is an important operating system for many of our customers, and we strive to stay in sync with its releases,” said Robert Dewar, AdaCore President and CEO. “GNAT Pro for LynxOS 5.0 will bring the most recent Ada technology - including new tools and libraries and improved code generation - to the latest version of LynxOS.”

“When we issue a new operating system release, customers expect that the tools they used previously - especially compilers and develop environments - will be available,” said Steve Blackman, LynxWorks’ Director of Business Development, Mil/Aero. “Ada is a language of choice for many LynxOS users, and we are pleased that AdaCore has ported their latest GNAT Pro development environment to LynxOS 5.0.”

About LynxWorks

LynxWorks, a world leader in the embedded software market, is committed to providing open and reliable real-time operating systems (RTOS) and software tools to embedded developers. The company’s LynxOS family of operating systems offers open standards with the highest level of safety and security features, enabling many mission-critical systems in defense, avionics and other industries. Additionally, LynxWorks’ BlueCat Linux provides the features and

support of embedded Linux for companies wanting to use open source technology for their embedded applications. The Eclipse-based Luminosity IDE gives a powerful and consistent development system across all LynxWorks operating systems. Since it was established in 1988, LynxWorks has created technology that has been successfully deployed in thousands of designs and millions of products made by leading communications, avionics, aerospace/defense, and consumer electronics companies. LynxWorks’ headquarters are located in San José, CA.

LynxWorks is a trademark and LynxOS and BlueCat are registered trademarks of LynxWorks, Inc. Linux is a registered trademark of Linus Torvalds. All other brand or product names are registered trademarks or trademarks of their respective holders.

AdaCore — GPS 4.4

From: AdaCore Press Center

Date: Tue, 13 Oct 2009

Subject: AdaCore Introduces Enhanced Version of GNAT Programming Studio

URL: <http://www.adacore.com/2009/10/13/gps-4-4/>

PARIS and NEW YORK, October 13, 2009 - AdaCore, a leading supplier of Ada tools and support services, today announced the release of GNAT Programming Studio (GPS) 4.4. This new version of AdaCore’s graphical Ada-oriented Integrated Development Environment (IDE) offers an improved user interface, faster performance and greater integration with AdaCore’s Project Coverage and SPARK Pro toolsets. GPS is provided with GNAT Pro on most platforms, for both native and embedded software development.

The most noticeable enhancement in GPS 4.4 is in the graphical user interface (GUI), which makes it easier for users to add new plug-ins and customize the IDE. The general navigation capabilities and documentation generation features have also been improved, ensuring consistent creation of supporting materials. And below the visible GUI, GPS now offers tighter integration with both the Project Coverage and SPARK Pro toolsets. Project Coverage’s code coverage and simulator capabilities can now be directly accessed from the IDE, and support for the SPARK language has been extended, allowing improved source/annotation navigation. SPARK developers may now more easily develop SPARK language applications and invoke the SPARK Pro toolset from GPS.

“We have been systematically enhancing the GNAT Pro toolset to improve support for safety-critical and high-security applications,” commented Arnaud Charlet, GPS Project Manager at

AdaCore. “GPS 4.4 is a good illustration of our strategy. Developers of high-assurance systems may now access Project Coverage and SPARK Pro tools with the same IDE that they already use for general development.”

Enhancements in GPS 4.4 include:

- Improved user interface look and feel
- Improved memory usage and speed
- New entity views
- Enhanced documentation generation to support both API documentation and source code browsing
- Hyperlinks added in source editor for quick source and web navigation
- Support for Project Coverage toolset invocation
- Improved SPARK Pro language and toolset support, in particular source navigation for annotations
- New “Tip of the Day” feature
- Support for filters in the locations display view
- Unified “visual diff” within the source editor
- Support for interleaved-Ada/Expanded-Ada (.dg files) in source editor
- Outline View updated in real-time, with the ability to display entities hierarchically
- New source navigation menus to display a type hierarchy
- New plug-ins, including:
 - o Formatting preferences that can be automatically set from gnatpp project switches
 - o Sources that can automatically be reformatted on save using gnatpp
 - o Simplified new file creation

GPS 4.4 is compatible with GNAT Pro versions 3.16a1 up to 6.3. As with all GNAT Pro components, GPS is distributed with full source code and is backed by AdaCore’s rapid and expert online support.

About GNAT Programming Studio (GPS)

GPS is a powerful Integrated Development Environment (IDE) written in Ada using the GtkAda toolkit. GPS’ extensive source-code navigation and analysis tools can generate a broad range of useful information, including call graphs, source dependencies, project organization, and complexity metrics. It also provides support for configuration management through an interface to third-party Version Control Systems, and is available on a variety of platforms including Altix Linux, IA64 HP Linux, Solaris (sparc and x86), GNU/Linux (x86 and x86-64), Mac OS X, and x86 Windows (2003, XP, Vista, and 7). GPS is highly extensible; a simple scripting

approach enables additional tool integration. It is also customizable, allowing programmers to specialize various aspects of the program's appearance in the editor for a user-specified look and feel.

Availability

GPS 4.4 is currently available as part of the GNAT Pro Ada Development Environment on selected platforms, and customers can download it via the GNAT Tracker tool. For the latest information on pricing and supported configurations please contact sales@adacore.com.

Webinar

A webinar focusing on the new features of the GPS 4.4 release will be presented on November 10, 2009 at 11:00 am (EST) / 5:00 pm (GMT).

For more information, or to register, please visit

<http://www.adacore.com/home/gnatpro/webinars/>

Aonix — ObjectAda for VxWorks/x86

From: Aonix Press Center

Date: Mon, 23 Nov 2009

Subject: Aonix Adds VxWorks/Intel® Architecture Target Support to ObjectAda Product Line

URL: http://www.aonix.com/pr_11_23_09.html

Company Set to Address Growing Demand for VxWorks/x86 Embedded Systems

San Diego, CA—November 23, 2009—Aonix®, a provider of solutions for safety and mission-critical applications, today announced the release of ObjectAda® Real-Time 8.4 for Windows, targeting Intel x86 architecture embedded and real-time systems running the Wind River VxWorks real-time operating system (RTOS). Following the release of ObjectAda Real-Time for VxWorks targeting PowerPC architectures earlier this year, this is the second ObjectAda Real-Time release supporting full Ada tasking atop VxWorks via Real-Time Processes (RTP).

ObjectAda Real-Time for Windows x Intel/VxWorks consists of a fully compliant ACATS 2.5 Ada 95 compiler plus supporting tools. It is compatible with Wind River's VxWorks environment, which comprises the VxWorks operating system and the VxWorks cross development toolset. ObjectAda for VxWorks leverages Wind River Workbench, an Eclipse-based development environment providing developers access to the broad range of tools available through the Eclipse framework. Users also have the option to utilize ObjectAda's standard graphical or command-line interface. The ObjectAda

compilation system is comprised of an integrated language-sensitive editor, source-code browser, compiler with industry-leading compilation speed, debugger and full library manager.

ObjectAda VxWorks provides runtime library support for execution of Ada on Intel Pentium targets via Wind River-supplied board support packages (BSPs), or for execution without target hardware via VxSim, a target simulation facility supplied in the VxWorks distribution. VxSim users can perform initial application execution and testing direct from the Intel/Windows host development platform.

"Aonix has provided Ada development and compilation tools for x86 chips for years," commented Gary Cato, Aonix director of marketing. "This ObjectAda product is the first support we've offered for VxWorks/Intel targets and is a welcome addition to the ObjectAda embedded, real-time and safety-critical line of products. With this product introduction Aonix now provides ObjectAda products for the full spectrum of Wind River RTOS products and enhances our ability as a Wind River Strategic Software Partner to provide additional platform options for our customers."

About the ObjectAda Family

ObjectAda is an extensive family of native and cross development tools and runtime environments. ObjectAda native products provide host development and execution support for the most popular environments including Windows, Linux and various UNIX operating systems. ObjectAda Real-Time products provide cross development tools on Windows, Linux or UNIX systems which target PowerPC and Intel target processors running in a full Ada "bare" runtime or in conjunction with popular RTOSs.

ObjectAda RAVEN® products provide a hard real-time Ada runtime to address those systems requiring certification to the highest levels of safety standards such as DO-178B Level A for flight safety.

Availability

ObjectAda Real-Time targeting the Intel x86 architecture running Wind Rivers' VxWorks is immediately available. Read more about ObjectAda Real-Time products.

About Aonix®

Aonix offers mission- and safety-critical solutions primarily to the military and aerospace, telecommunications and transportation industries. Aonix delivers the leading highly reliable, real-time embedded virtual machine solution for running Java™ programs deployed today and has the largest number of certified Ada applications at the highest level of criticality. Headquartered in San Diego,

CA and Paris, France, Aonix operates sales offices throughout North America and Europe in addition to offering a network of international distributors.

For more information, visit www.aonix.com.

Lattix — Lattix 5.5

From: Lattix Press Center

Date: Tue, 10 Nov 2009

Subject: Lattix Releases Lattix 5.5

URL: <http://www.lattix.com/node/120>

Award-winning software architecture management solution provides new API and scripting capabilities to extend and integrate Lattix with more platforms and tools.

Boston, MA - November 10, 2009- Lattix Inc., a leading provider of innovative software architecture management solutions, today announced the release of its newest solution, Lattix 5.5. This solution includes powerful new functionality to enable architects, developers and managers to identify issues, re-engineer, and measure the quality of complex software systems.

In addition to performance improvements and feature enhancements, Lattix 5.5 provides a new API which can be used to extend and customize the functionality of Lattix. Also new in Lattix 5.5 are improved integrations to Microsoft Visual Studio and Klocwork, as well as new integrations with the popular graphing packages GraphViz and Pajek.

"The new API enables our users to customize Lattix to fit seamlessly in their development environment," explains Neeraj Sangal, president and founder of Lattix.

"With a number of scripts already available from Lattix, our customers can now create more precise models and achieve better results through higher utilization."

Users can now write their own Groovy scripts to manipulate and query a Lattix project, or change an existing script to meet their requirements.

The Scripts Repository in the Lattix KnowledgeBase provides access to scripts such as those which:

- enable integration with build systems for change impact analysis
- display a variety of network diagrams using GraphViz
- generate a list of dependency paths between two elements

The new integration with Microsoft Visual Studio enables the source code for a selected dependency or element definition to be displayed in Visual Studio.

With Lattix LDC, it is possible to automatically update the Lattix project

with source file and line number information as part of the Visual Studio build process.

The DSM, CAD, and GraphViz visualizations of Lattix 5.5

About Lattix 5.5

Lattix 5.5 provides the most comprehensive solution for systems that include codebases, databases, frameworks, and UML/SysML models. Lattix 5.5 supports XMI and IBM Rational Rhapsody models; Ada, C/C++, Java, .NET, and Pascal languages; Oracle, SQL Server, and Sybase databases; and Spring and Hibernate frameworks. Lattix 5.5 also provides support for full web-based reporting of architectural metrics, violations, and incremental changes.

To learn more about Lattix 5.5 and explore the different solutions that are available, please visit <http://www.lattix.com/products>.

Lattix 5.5 enables companies to improve and maintain quality, lower defect rates, enhance testability, lower costs through more effective development, and manage risks by better understanding of the impact of proposed changes.

Availability

Lattix 5.5 is available immediately from Lattix in the US or from our partners throughout Europe, the Middle East, and Asia Pacific. A variety of license options are available, from individual user to enterprise floating licenses. A free evaluation license is also available for download from

<http://www.lattix.com/gettingstarted>.

About Lattix

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

Lattix provides a powerful new approach of utilizing dependency models for automated analysis and enforcement of architectures. Lattix is located in Andover, MA. More information about Lattix can be found at www.lattix.com.

Ada and GNU/Linux

Debian Policy for Ada

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 26 Oct 2009 11:54:27 -0700 PDT
Subject: ANN: Debian Policy for Ada, Fourth Edition for Debian 6.0 "Squeeze"
Newsgroups: comp.lang.ada

Stephe Leake and I are now satisfied with our latest changes to the Debian Policy for Ada. I have now published it[1] and will commence the transition of all Ada package to this new policy; this includes

moving to gnat-4.4 as the Ada compiler and adding "aliversion" numbers to all -dev package names.

[1] <http://people.debian.org/~lbrenta/debian-ada-policy.html>

<http://people.debian.org/~lbrenta/debian-ada-policy.pdf>

<http://people.debian.org/~lbrenta/debian-ada-policy.txt>

<http://people.debian.org/~lbrenta/debian-ada-policy.info>

<http://people.debian.org/~lbrenta/debian-ada-policy.texi>

This new policy applies to the release of Debian currently in development, 6.0 "Squeeze". The policy for the current release, 5.0 "Etch", is still available[2].

[2] <http://people.debian.org/~lbrenta/5.0-lenny/debian-ada-policy.html>

<http://people.debian.org/~lbrenta/5.0-lenny/debian-ada-policy.pdf>

<http://people.debian.org/~lbrenta/5.0-lenny/debian-ada-policy.txt>

<http://people.debian.org/~lbrenta/5.0-lenny/debian-ada-policy.info>

<http://people.debian.org/~lbrenta/5.0-lenny/debian-ada-policy.texi>

[In the original thread, the author unfortunately provided incorrect links for [2], as noted by Niklas Holsti. In a subsequent thread by Ludovic Brenta the correct links were published. The links reported herein are already in their correct form. —mp]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Date: Sun, 15 Nov 2009 02:23:32 -0800 PST

Subject: The Debian Policy for Ada is now official

Newsgroups: comp.lang.ada

The Debian Policy for Ada has received the blessing of the Debian organization at large; a link to the document now appears on the Debian Developer's Corner along with all other policy documents:

<http://www.debian.org/devel/>

GNAT and Maemo

From: Michael Bode <m.g.bode@web.de>
Date: Sun, 15 Nov 2009 17:37:28 +0100
Subject: Gnat and Maemo
Newsgroups: comp.lang.ada

How difficult would it be to start programming in Ada for the Maemo platform? I understand one would need a cross compiler to the armel target and the GNAT runtime for armel and somehow include that in the (Debian-based) Maemo SDK. Is anyone working on this?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Date: Sun, 15 Nov 2009 14:56:47 -0800 PST

Subject: Re: Gnat and Maemo
Newsgroups: comp.lang.ada

[...]

Laurent Guerby and others are working on enhancing GNAT so that it supports arm and armel as a target. Unfortunately, only SJLJ exception handling works at the moment and SJLJ is both very slow and non-standard on that architecture; the EABI mandates ZCX. See the threads stating at:

<http://gcc.gnu.org/ml/gcc/2009-08/msg00192.html>

<http://gcc.gnu.org/ml/gcc-patches/2009-09/msg00450.html>

It is not ready for use yet. I'm sure they'll welcome help.

[...]

PragmAda Reusable Components in Fedora

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>

Date: Tue, 10 Nov 2009 22:46:49 +0100

Subject: Doubling the number of Ada libraries in Fedora

Newsgroups: comp.lang.ada

The PragmAda Reusable Components library is currently being distributed as an update to Fedora 11, and also to Fedora 12 (which is scheduled to be released in a week). As GTKada is also available, this means that there are now a whopping *two* Ada libraries in Fedora! Well, there's still a long way to go but this is at least a start.

Once the mirrors have caught up, the library can be installed with «yum install PragmARC-devel», and using it is as simple as «with "pragmarc";» if you use GNAT project files.

Ada and Microsoft

Linking with GNAT on Windows

From: Maciej Sobczak <maciej@msobczak.com>

Date: Tue, 24 Nov 2009 08:26:00 -0800 PST

Subject: Linking with GNAT on Windows
Newsgroups: comp.lang.ada

[...]

Imagine a C library compiled with Visual Studio. This library is used by an Ada program by means of pragma Import.

When the C library is compiled in the Debug mode, GNAT can link the whole program. If the same C library is compiled in the Release mode, the GNAT linker says "undefined reference" with regard to the C function that is imported by Ada.

I am pretty sure that people who are regularly targeting the Windows platform have seen that already and could possibly help with a hint (compiler/linker option on the Visual Studio side?) on how to proceed.

All suggestions are welcome.

This is GNAT 2009 and Visual Studio 2008.

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Tue, 24 Nov 2009 18:34:43 +0100

Subject: Re: Linking with GNAT on Windows

Newsgroups: comp.lang.ada

[...]

Hmm, it is no matter how the program is compiled, but how is it linked.

Debug/Release are just project scenario names, which can mean anything in the concrete project.

Of course we cannot exclude that some inspired C programmer could make use of `#ifdef __DEBUG__` to change names of all functions in the source...

AFAIK, GNAT does not recognize the MS lib files [...]

There also exist *.def files which may influence the names of the entries in the import library.

Further there exist `__declspec(dllexport)`, `__cdecl`, `__stdcall` modifiers in the program, which might have effect on the external names.

Plus in Visual Studio there can be defined post build steps. which might call scripts and do, well, anything.

All in one, it is impossible to say what is going on. You have to verify all steps. I would ensure that *.a file is created and used by GNAT. Then I would check the names in it (nm -s). Then I would verify the content of its source *.lib file etc.

From: Kevin K <kevink4@gmail.com>

Date: Tue, 24 Nov 2009 17:36:06 -0800

PST

Subject: Re: Linking with GNAT on Windows

Newsgroups: comp.lang.ada

What I ended up doing when I needed to link some C code compiled under Visual C++ into an Ada main (due to the code using APIs that weren't supported under the SDK in GNAT environment) was to build a DLL with the appropriate declaration. The downside is that you can't debug it that way. When debugging is necessary, I need to create a driver in C within Visual C++ and debug it there.

From: Maciej Sobczak

<maciej@msobczak.com>

Date: Wed, 25 Nov 2009 00:43:11 -0800

PST

Subject: Re: Linking with GNAT on Windows

Newsgroups: comp.lang.ada

[...]

> Hmm, it is no matter how the program is compiled, but how is it linked.

The GNAT invocation is the same in both cases and includes this:

`-largS -lmylibrary`

with `mylibrary.lib` (the result of compilation on Visual Studio) file somewhere around.

> Of course we cannot exclude that some inspired C programmer

I was that programmer. The whole C library amounts to a single function with single line of code ("Hello from C", essentially), no preprocessor and no other tricks. The function is declared as extern "C" to avoid name mangling.

> AFAIK, GNAT does not recognize the MS lib files

It seems to recognize them. Not only it complains when the file is not around (note: it can automatically make an association between `-lmylibrary` linker option and the `mylibrary.lib` file - this would not be the case if .lib files were not supported at all), but it really works fine if the C library is build in the Debug mode.

I have tried to analyze all options in these two modes, but do not see any differences that would affect this.

> There is also exist *.def files which may influence the names of the entries in the import library.

Yes, but this is not used. My naive first diagnostics was that the library compiled in Debug mode has its names exported by default, whereas the Release mode would need the .def file. This theory is contradicted by the fact that a test C program can use that library no matter how it was compiled. But then, it is the single toolchain on the whole path.

> Further there exist `__declspec(dllexport)`, `__cdecl`, `__stdcall` modifiers in the program, which might have effect on the external names.

None of these are used. Note that it is a static library, not a DLL.

> Plus in Visual Studio there can be defined post build steps.

These are not defined. The C library was created as a pristine project.

> All in one, it is impossible to say what is going on.

Cool. I am pretty convinced that this is not even a GNAT issue, but rather concerns the interaction of Visual Studio and MinGW toolchain that is a back-end for GNAT.

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Wed, 25 Nov 2009 10:46:56 +0100

Subject: Re: Linking with GNAT on Windows

Newsgroups: comp.lang.ada

[...]

Usually debug/release versions have different names like `mylibraryd.lib` or else placed into different subdirectory like `.Debug` vs. `.Release`. What about the "-L" switch?

> [...] The function is declared as extern "C" to avoid name mangling.

OK, extern "C" normally mangles, e.g. adds underscore in front of the name.

Also, if I correctly remember, `stdcall` convention adds some funny suffixes like "@4" to the names in import libraries. But that behavior is independent on debug/release.

[...]

From: Maciej Sobczak

<maciej@msobczak.com>

Date: Wed, 25 Nov 2009 04:47:24 -0800

PST

Subject: Re: Linking with GNAT on Windows

Newsgroups: comp.lang.ada

[...]

> Usually debug/release versions have different names like `mylibraryd.lib` or else placed into different subdirectory like `.Debug` vs. `.Release`. What about the "-L" switch?

If the file was not found, the linker would say so. Every `-lmylibrary` option must have a corresponding library file.

[...]

As already said, this is the interaction between Visual Studio and MinGW, so can be completely reproduced outside of GNAT.

The problem proved to be related to the Visual optimization option that is used to enable so-called whole-program optimization (Enable link-time code generation). On command-line this is `/GL` option.

This option must be switched off. I hope this observation will help other programmers.

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Wed, 25 Nov 2009 14:44:26 +0100

Subject: Re: Linking with GNAT on Windows

Newsgroups: comp.lang.ada

[...]

MSDN:

".obj files produced with `/GL` and precompiled header files should not be used to build a .lib file unless the .lib file will be linked on the same machine that produced the `/GL` .obj file. Information from the .obj file's precompiled header file will be needed at link time."

That explains a lot. Noteworthy is mentioning precompiled headers. That stuff never ever worked in VC... Presumably /GL silently killed the entry point in the .lib file.

References to Publications

Embedded Systems Design — "Software for dependable systems"

From: Peter Hermann

Date: Fri, 20 Nov 2009 12:52:09 +0000 UTC

*Subject: Software for dependable systems
By Jack Ganssle in Embedded Systems Design ESD*

Newsgroups: comp.lang.ada

great article:

Software for dependable systems

by Jack Ganssle

in Embedded Systems Design ESD
vol22,#10,Nov2009, page 37

or click on News20091120 on top of

http://www.ihr.uni-stuttgart.de/forschung/ada/resources_on_ada/

[read the article at:

<http://www.embedded.com/columns/breakpoint/220900315—mp>]

Ada Inside

Praxis HIS — SPARK used for the AgustaWestland AW159

From: Praxis HIS News Center

Date: Thu, 3 Sep 2009

Subject: General Dynamics UK selects SPARK language for major new Royal Navy helicopter project

URL: <http://www.praxis-his.com/news/generalDynamics.asp>

SPARK used to develop safety-critical system for AgustaWestland AW159 Lynx Wildcat.

Praxis, the international specialist in critical systems engineering and assurance, today announced that General Dynamics UK has selected Praxis' SPARK language as part of its £6 million contract to develop the safety-critical Stores Management System for the Royal Navy's new AgustaWestland AW159 Lynx Wildcat helicopter.

SPARK is a high level programming language and toolset designed for writing software for high integrity applications. General Dynamics UK selected SPARK for the project owing to its ability to enable the development and verification of software to the highest level of

Ministry of Defence safety certification – Defence Standard (Def Stan) 00-56 issue 2 Safety Integrity Level 4. SPARK enables the application of formal verification techniques in a segregated monitor architecture, ensuring rapid compliance.

Up to seven developers will use SPARK to create the Stores Management System, which controls the deployment of weaponry from the AW159 Lynx Wildcat.

Project development completes in mid 2011 and will cover more than 40,000 lines of SPARK code.

The AW159 Lynx Wildcat (formerly called the Future Lynx) will be the Royal Navy's new maritime surveillance and attack helicopter. Scheduled to enter service in 2015, 28 helicopters have been ordered for the Royal Navy. The AW159 will provide ship defence against surface threats, act in an anti-submarine role and operate as a light utility helicopter. The Stores Management System will enable the AW159 to operate the lightweight Sting Ray torpedo as well as the anticipated Future Air-to-Surface Guided Weapon (FASGW).

General Dynamics UK has used SPARK across its Stores Management System product line since the early 1990s. Previous projects include systems for the Tornado, Harrier and Typhoon aircraft.

"Meeting strict safety-critical certification is central to the new Stores Management System for the AW159 Wildcat," said Steve Hewitt, Programme Manager, Mission & Security Systems, General Dynamics UK Limited. "Our ongoing partnership with Praxis meant that SPARK was the natural choice when it came to developing this mission-critical application to the highest safety standards."

Developed by Praxis, SPARK is a language specifically designed to support the development of software used in applications where correct operation is vital either for reasons of safety or security. The SPARK Toolset offers static verification that is unrivalled in terms of its soundness, low false-alarm rate, depth and efficiency. The toolset also generates evidence for correctness that can be used to build a constructive assurance case in line with the requirements of industry regulators and certification schemes.

"General Dynamics UK is an extremely mature user of Praxis' SPARK technology, and I am delighted that they have once again chosen our technology for a new development project," said Keith Williams, Praxis Managing Director. "This secures SPARK as a core tool for assuring the integrity of advanced weapons systems with its use on a wide range of air platforms."

About Praxis

Praxis is a systems engineering company specialising in safety and mission critical applications. Praxis leads the world in specific areas of advanced systems engineering such as: ultra low defect software engineering, safety engineering for complex or novel systems, and tools/methods for systems engineering. Praxis offers clients a range of services including turnkey systems development, consultancy, training and R&D. Key market sectors are Aerospace, Defence, Air Traffic Management, Railways and Nuclear. The company operates internationally with active projects in the US, Asia and Europe.

The headquarters of Praxis are in Bath (UK) with offices also in London, Loughborough and Paris. It is wholly owned by Altran Technologies which is a global leader in innovation engineering and employs 18,500 staff across the world.

www.praxis-his.com

About General Dynamics

General Dynamics United Kingdom Limited, a wholly owned subsidiary of General Dynamics (NYSE: GD), is a leading player in the UK's knowledge economy and industrial base. Established in the United Kingdom for over 40 years, it employs over 1,600 people at 10 UK and international facilities. A prime contractor and complex systems integrator, working in partnership with the Ministry of Defence (MoD) and other allies, growing key intellectual property, skills and capabilities in its UK research facilities and workforce, whilst harnessing world-leading technology.

General Dynamics UK led a key MoD Defence Technology Centre research consortium and, together with a growing C4I export programme, plays a central role manufacturing and developing technology to deliver network enabled capability and ISTAR in the battlespace. The Company is widely recognised as a leading contender to supply and integrate the next generation of Armoured Fighting Vehicles for the British Army. For further information visit

www.generaldynamics.uk.com

General Dynamics, headquartered in Falls Church, Va., employs approximately 92,000 people worldwide. The company is a market leader in business aviation; land and expeditionary combat systems, armaments and munitions; shipbuilding and marine systems; and information systems and technologies.

More information about General Dynamics is available online at www.gd.com

Use of SPARK at Genode labs

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Date: Thu, 26 Nov 2009 01:18:56 -0800 PST

Subject: Genode implements zero-footprint runtime for Ada and SPARK

Newsgroups: comp.lang.ada

Today on osnews.com's front page an innocuous phrase caught my attention so I followed the links...

<http://genode.org/documentation/release-notes/9.11#section-19>

"At Genode Labs, we are exploring the use of the SPARK subset of Ada to implement security-critical code and use Genode as development platform. For this reason, we have added support for executing freestanding Ada code on Genode."

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Date: Thu, 26 Nov 2009 07:45:51 -0800 PST

Subject: Re: Genode implements zero-footprint runtime for Ada and SPARK

Newsgroups: comp.lang.ada

[...]

> Hm, later on this page it says "Elaboration is not performed". How can then Ada code work at all?

By restricting the compilation units in the program to only Pure and Preelaborated units, I suppose.

From: Norman Feske

<norman.feske@genode-labs.com>
Date: Thu, 26 Nov 2009 07:47:59 -0800 PST

Subject: Re: Genode implements zero-footprint runtime for Ada and SPARK

Newsgroups: comp.lang.ada

Hello,

as a developer of Genode, I am happy about the response to our Ada-related addition. As stated in the release notes, the current integration of Ada support is mainly geared towards using Genode as an experimentation platform for developing SPARK sub programs. It is just the first step. If more people outside the current Genode developer community show interest in this particular topic, we will be happy to extend the Ada support as needed.

> Hm, later on this page it says "Elaboration is not performed". How can then Ada code work at all?

The mentioned limitation refers to package initializations normally performed by the startup code generated by gnatbind, specifically the 'ada_init' function.

The current use case for Ada on Genode is to use SPARK for creating type and

utility packages with free-standings functions called from C code.

Because such packages have no internal state and no begin-end block, we can omit gnatbind for now. However, should the need for elaborating packages in the right order arise in the future, support for calling gnatbind will be added to the build environment. In short, the solution comes down to calling gnatmake with the '-c -b' arguments, adding the generated startup code to the program, and actually calling 'adainit'.

[...]

Indirect Information on Ada Usage

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

Job offer [Belgium]: Ada Programmer

Ada Programmer with French and English language skills is required for a 6 month contract in the south of Belgium. The role will be to develop basic signalling application and monitor software as well as integration and functional testing. Candidates will need a knowledge of Ada, functional software specification (Teamwork tool), embedded software and safety protocols.

Job offer [United Kingdom]: Ada SPARK software engineer

[...] Ada software engineer with SPARK experience [...].

The ideal candidate will be responsible for software development of DO178B software. [...]

Skills and experience:

- Software engineer with Ada SPARK experience
- DO178B software development experience
- Experience of Artisan UML design
- Development experience of Aero engine control software FADEC including fuel pumps and fuel driven actuators

Job offer [United Kingdom]: Graduate Spacecraft SW Engineer

Graduate / Junior Spacecraft Simulations SW Engineer - Maths/Physics Degree C++/Java/Fortran/Ada

Tasks:

- Developing spacecraft simulators used to train spacecraft operators

Essential Skills

- 1st Class or 2:1 degree in Mathematics/Physics/Engineering subject
- Knowledge of spacecraft systems
- Proficiency in C or C++ or Java or Fortran or Ada

- Fluency in English

Desirable Skills

- Proficiency in C++
- Knowledge of software development standards and methods
- Knowledge of orbital mechanics
- Knowledge of control system theory
- Experience of using processor emulators in simulations

Job offer [United Kingdom]: Software Engineer

[...] Requires familiarity with object oriented design methodologies, UML and experience of software development generally. Desirable experience includes: Ada 95 development, experience with iData or equivalents (eg. VAPS), and CORBA development. The Software Engineer appointed will be responsible for design, development, analysis, testing, and documentation of a complex mission-critical computing system. The successful Software Engineer will be joining a large and highly successful multinational.

[...]

Key responsibilities of the Software Engineer:

- Design, development and unit test of elements of software.
- General supervision in planning and control of own work.
- Design using object oriented methodologies and a UML toolset (Artisan Studio).
- Code development in Ada 95 (using AdaMulti).
- Unit testing (using AdaTest).
- Correct program errors, prepare operating instructions, compile documentation of program development, and analyse system capabilities to resolve questions of program intent, output requirements, input data acquisition, programming techniques, and controls.
- Assignment to other teams may be required.

[...]

Job offer [Italy]:

[...]

Development of software under RTCA DO-178B, using Ada and/or C and targeting embedded safety-critical applications.

The development environment is Green Hills AdaMULTI 2000.

The design methodology is HOOD.

Other software used during the software development:

- PVCS Dimensions 7.2
- DOORS v8.0

- TNI Stood

Software development is supported with the following tools for debugging/integration on target:

- Green Hills AdaMULTI 2000 Debugger
- Power PC Probe (JTAG Connections)
- A proprietary engineering test bench-Lab equipment (Sampling Oscilloscope, Waveform Generator, Dynamic Signal Analyzer)

Microsoft Office 2000 is used to support the production of documentation.

[...]

Educational background:

Technical diploma with at least 15 years' work experience or Laurea specialistica [Master's degree —mp] in Computer Science / Computer Engineering.

Knowledge as software developer of the following operating systems:

- Windows 2000/XP/.Net
- UNIX

Knowledge of the following programming languages:

- Ada, C, Assembler

[...]

[Translated from Italian —mp]

Ada in Context

Dynamic allocation of unconstrained types

From: Maciej Sobczak

<maciej@msobczak.com>

Date: Wed, 30 Sep 2009 07:29:26 -0700

PDT

Subject: Dynamic allocation of unconstrained types

Newsgroups: comp.lang.ada

Consider:

procedure Test is

package P is

type T (<>) is limited private;

function Create return T;

private

type T is limited record

l : Integer;

end record;

end P;

package body P is

function Create return T is

begin

return T(l => 123);

end Create;

end P;

S : **access** P.T;

begin

S := **new** P.T'(P.Create); -- ??? (this is line 22)

end Test;

GNAT says:

test.adb:22:19: uninitialized unconstrained allocation not allowed

test.adb:22:19: qualified expression required

Interestingly, it works with Strings.

Why doesn't GNAT recognize it as a qualified expression?

I would like to allocate dynamically something that has a constructor function. There is no other way to create the object than with that function and presumably it should be possible to use it with dynamic allocation. How can I do it?

BTW - when preparing this example I tried first with empty (null) record, but got stuck with proper way to return an instance of T. I remember there was some older discussion about it, but for some reason I cannot find it and the following:

```
return T'(others => <>);
```

is rejected as well.

What is the proper way to create null aggregates?

From: Robert A Duff

<bobduff@shell01.TheWorld.com>

Date: Wed, 30 Sep 2009 10:50:05 -0400

Subject: Re: Dynamic allocation of unconstrained types

Newsgroups: comp.lang.ada

I try to avoid the use of anonymous access types.

They cause too many surprises.

> begin

```
S := new P.T'(P.Create); -- ??? (this is line 22)
```

end Test;

GNAT says:

test.adb:22:19: uninitialized unconstrained allocation not allowed

test.adb:22:19: qualified expression required

Looks like a bug in the compiler.

[...]

return (null record);

or

return T'(null record);

The "others => <>" should work, too.

[...]

From: Adam Benesch

<adam@irvine.com>

Date: Wed, 30 Sep 2009 07:54:53 -0700

PDT

Subject: Re: Dynamic allocation of unconstrained types

Newsgroups: comp.lang.ada

[...]

> Interestingly, it works with Strings.

Why doesn't GNAT recognize it as a qualified expression?

It looks like a bug to me. I do notice that if you remove "limited" from both declarations of T, then it works.

There are some new Ada 2005 rules that allow limited-type expressions in more places (functions returning limited types weren't allowed in Ada 95), so it's likely that there was a mistake in implementing this new feature.

[...]

Try

return T'(null record);

The first attempt, with (others => <>), should have worked, but there was faulty language in the RM that made this illegal for a null record. It has been fixed by a Binding Interpretation, AI05-16, so I think compilers should allow that construct, but perhaps it hasn't yet been fixed in the version of GNAT you're using.

From: Jeffrey R. Carter

<jrcarter@acm.org>

Date: Wed, 30 Sep 2009 11:30:28 -0700

Subject: Re: Dynamic allocation of unconstrained types

Newsgroups: comp.lang.ada

[...]

> (functions returning limited types weren't allowed in Ada 95)

Yes, they were. The current revision changed them significantly.

From: Adam Benesch

<adam@irvine.com>

Date: Wed, 30 Sep 2009 12:15:22 -0700

PDT

Subject: Re: Dynamic allocation of unconstrained types

Newsgroups: comp.lang.ada

[...]

Oh, yeah, that's right---the return-by-reference thing. I forgot about that because they were really more like functions that return an access to some existing object, rather than functions returning an object. You couldn't create a new limited object and return it. Anyway, that doesn't really affect my point: the compiler's implementation of functions returning limited types is pretty much all new for Ada 2005, and this appears to be a case that slipped through the cracks in GNAT.

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Wed, 30 Sep 2009 16:56:41 +0200

Subject: Re: Dynamic allocation of unconstrained types

Newsgroups: comp.lang.ada

> Interestingly, it works with Strings.

Because String is not a limited type.

> Why doesn't GNAT recognize it as a qualified expression?

Qualified expression is not allowed for a limited types, logically. (Not yet, I think it will be a necessary step in order to continue the idea of limited aggregates. Once we allowed them, there is no reason not to allow qualified aggregates and thus limited expressions. After all aggregate is an expression. Language design bugs are always punished in the end...)

> I would like to allocate dynamically something that has a constructor function. There is no other way to create the object than with that function and presumably it should be possible to use it with dynamic allocation.

> How can I do it?

By providing a function that explicitly returns an access to T.

What you are trying to do: is to enforce a custom initialization on a limited private type. That does not work in Ada, alas. It is hopeless, I am afraid. The best way I know is to make it public, removing the indefinite constraint. Otherwise you will get mounting problems crippling your design more and more, and still get no working solution.

> What is the proper way to create null aggregates?

return (null record);

if you need to specify the type, like when the formal result is class-wide, then:

return X : T;

*From: Adam Benesch
<adam@irvine.com>*

*Date: Wed, 30 Sep 2009 08:03:57 -0700
PDT*

*Subject: Re: Dynamic allocation of
unconstrained types*

Newsgroups: comp.lang.ada

> Qualified expression is not allowed for a limited types, logically.

Yes, they are; see 7.5(2.1-2.9).

On task deallocation

*From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>*

Date: Tue, 17 Nov 2009 11:17:38 +0100

*Subject: Tail recursion upon task
destruction*

Newsgroups: comp.lang.ada

Consider a task encapsulated in an object in either way:

type Device is

 Driver : Driver_Task (Device'Access);

or

type Device is

 Driver : **not null access** Driver_Task
 := Driver_Task (Device'Access);

Let the object is allocated dynamically and we wanted to destroy it from the task. It seems that there is no way to do this:

```
task Driver_Task (
  Object : not null access Device) is
procedure Free is
  new Ada.Unchecked_Deallocation
    (Device, Device_Ptr)
  Self : Device_Ptr;
begin
  ...
accept Shut_Down;
  Self := Object.all'Unchecked_Access;
  -- Or whatever way
  Free (Self); -- This will deadlock
end Driver_Task;
```

The core problem is that a task cannot destroy itself, because that would block for task termination, which never to happen.

What I do to solve this is an extra "collector task" to await for a rendezvous with Driver_Tasks, accepting a pointer to the Device and then after leaving the rendezvous, freeing it. That looks tedious.

Don't we need some kind of "tail recursion" for this destruction pattern?

*From: Randy Brukardt
<randy@rsoftware.com>*

Date: Tue, 17 Nov 2009 15:38:45 -0600

*Subject: Re: Tail recursion upon task
destruction*

Newsgroups: comp.lang.ada

[...]

Ada does not allow an object to destroy/free itself. That's generally a good thing, because such an object cannot be an ADT (it cannot be used as the element of a container, for instance), and such a model would require a far more complex scheme of frame completion than is used now: wait for all tasks, then finalize all objects, then free all memory.

I realize that there are a few cases where some other scheme would be better (we struggled with this in CLAW, as the finalization of library level objects tried to use the GUI task which of course has already terminated), but they would require such an earthquake in semantics as not to make any sense for Ada.

In your particular case, I don't understand why you don't use nesting to solve the problem. That is, put the object inside of the task (either directly or logically), so it can be destroyed when the task needs to do that.

That would look something like:

```
task type Device;
task type Device is
  Device_Data : not null access
  Device_Data_Type := new
  Device_Data_Type;
begin
  ...
  Free (Device_Data);
```

end Device;

Note: you'd need a named access type to actually do this - I used an anonymous one simply to make my point clearer.

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Wed, 18 Nov 2009 09:41:01 +0100

*Subject: Re: Tail recursion upon task
destruction*

Newsgroups: comp.lang.ada

[...]

> Note: you'd need a named access type to actually do this - I used an anonymous one simply to make my point clearer.

Unfortunately that does not work. I simplified the task description.

Actually in my case the device has some associated objects, say, screws. They are reference counted, both the devices and screws. A device screw holds a reference to its device. The screws are used somewhere in the application. You can create and remove screws and devices. The application may hold references them.

Now consider a case when the last screw is removed from the device. This is an operation eventually serviced by the device driver. I.e. within the device driver, you see, it was the last screw of the device and *if* there is no other references to the device, it must fall apart. This is a case where you wanted the device to commit suicide. There is nobody else out there to do this. The device is dangling. This is not the only use case, just one possible case.

And, considering the design. It looks logical that if screws are ultimately removed at some dedicated context (of the device driver), then the devices themselves could also be removed on the context of some "collector task".

Nevertheless, I am not sure that all cases where active objects should "commit suicide", should/could be treated this way.

From: Georg Bauhaus

Date: Wed, 18 Nov 2009 12:02:34 +0100

*Subject: Re: Tail recursion upon task
destruction*

Newsgroups: comp.lang.ada

[...]

Could you make a Hammer task that will perform its duties whenever a Device is reported/reports to have lost all its screws? (Yes, a garbage collector, I think, though explicitly co-operating with devices.)

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Wed, 18 Nov 2009 14:29:13 +0100

*Subject: Re: Tail recursion upon task
destruction*

Newsgroups: comp.lang.ada

[...]

Yes, this is what I did.

But the question is of the general nature, why there should be an extra task to destroy the given one? So the argument should be also general, like the Randy's one about ADTs.

The counter argument and the problem is that the relation between an object and its task is not evident in Ada, for multiple reasons. One of them is that tasks are not tagged. So there is a problem, because when this relation is ignored or missed by the designer, then a straightforward implementation of the task will sometimes deadlock. That is not good.

From: Stefan Lucks <stefan.lucks@uni-weimar.de>

Date: Wed, 18 Nov 2009 11:31:59 +0100

Subject: Re: Tail recursion upon task destruction

Newsgroups: comp.lang.ada

> Now consider a case when the last screw is removed from the device. This is an operation eventually serviced by the device driver. I.e. within the device driver, you see, it was the last screw of the device and *if* there is no other references to the device, it must fall apart. This is a case where you wanted the device to commit suicide. There is nobody else out there to do this. The device is dangling. This is not the only use case, just one possible case.

OK, so you have a task (a device) which notices that it is no longer useful. You would like such a task to do some cleanup and to commit "suicide". Unfortunately, it can't do the cleanup after the "suicide", because it is "dead" then. And it can't cleanup itself before being "dead" because it needs its local memory until the very moment of its "death".

But couldn't you just use (or maybe abuse) the features from Ada.Task_Termination to do perform the cleanup, after the task has died? Even if the "death" is not by suicide (apart from "suicide" = regular termination, the options are "murder" = abort and "accident" = unhandled exception).

See <http://www.adaic.org/standards/05rat/html/Rat-5-2.html#I1150>.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>

Date: Wed, 18 Nov 2009 18:48:06 +0100

Subject: Re: Tail recursion upon task destruction

Newsgroups: comp.lang.ada

[...]

Yes, it is an interesting option. One could terminate the task and from the handler kill the object. The difficulty is that Ada.Task_Termination is not generic. It is not possible to pass a reference to the object to the handler.

From: Egil Høvik

<egilhovik@hotmail.com>

Date: Thu, 19 Nov 2009 01:25:25 -0800

PST

Subject: Re: Tail recursion upon task destruction

Newsgroups: comp.lang.ada

This would still be a bounded error.

A task is not terminated until the task body has been finalized RM-9.3(5), and since Task_Termination handlers are executed as part of the finalization of task bodies RM-C.7.3(14/2), you would violate RM-13.11.2(11) by deallocating the task in the handler.

Unchecked_Deallocation of class-wide objects

From: Maciej Sobczak

<maciej@msobczak.com>

Date: Mon, 28 Sep 2009 01:43:03 -0700

PDT

Subject: Unchecked_Deallocation of class-wide objects

Newsgroups: comp.lang.ada

Is it legal and safe to deallocate class-wide objects?

The problem is that such an object is allocated with its concrete type, whereas deallocation is defined for its class-wide type.

Consider:

```
type Shape is tagged private;
```

```
type Shape_Access is access  
Shape'Class;
```

```
procedure Free_Shape is new  
Ada.Unchecked_Deallocation  
(Object => Shape'Class,  
Name => Shape_Access);
```

```
-- ...
```

```
type Circle is new Shape with ...
```

```
-- ...
```

```
C : Shape_Access := new Circle;
```

```
-- ...
```

```
Free_Shape (C);
```

Is the Circle object allocated on the Shape-wide storage pool? From what I understand, this is the condition for the above to work properly. What if Circle is allocated for some Circle_Access type which is then converted to Shape_Access? Can it be safely deallocated?

I believe that the above is a pretty standard use-case, but I would like to confirm that. Unfortunately, AARM is not very explicit about this subject.

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Mon, 28 Sep 2009 11:12:04 +0200

Subject: Re: Unchecked_Deallocation of class-wide objects

Newsgroups: comp.lang.ada

[...]

Yes, deallocation "dispatches" on the pointer's target.

[...]

In Ada pool is bound to the access type, not to the target type, which is logical consequence that an object can be allocated on the stack.

Another consequence is that it is meaningless to talk about Shape-wide-pool, however an implementation may indeed allocate objects of different types in different pools transparently to the program. If it chooses to do this for tagged types of the same hierarchy, then the pointer should become fat and contain the type tag in it. I know no Ada compiler that does it this way, but it is a possible scheme, IMO.

> What if Circle is allocated for some Circle_Access type which is then converted to Shape_Access? Can it be safely deallocated?

You could not convert it because Shape_Access is pool-specific.

(Unchecked_Conversion tells for itself)

If it were a general access to class wide then deallocator would be "doubly dispatching" on the pool and on the target. Thus, as far as I can tell, it is safe in both cases.

On dispatching calls in Ada and C++

From: Markus Schoepflin

<markus.schoepflin@comsoft.de>

Date: Fri, 20 Nov 2009 14:15:33 +0100

Subject: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

[...]

I'm trying to fell may way around object oriented Ada programming, and I think I must be missing something absolutely basic. Please consider the following package:

```
package FOOS is
```

```
  type FOO is abstract tagged
```

```
  null record;
```

```
  procedure P (THIS : in FOO);
```

```
  procedure A (THIS : in FOO)
```

```
  is abstract;
```

```
end FOOS;
```

```
package body FOOS is
```

```
  procedure P (THIS : in FOO) is
```

```
  begin
```

```
    A (THIS);
```

```
  end;
```

```
end FOOS;
```

When trying to compile this, I get:

```
foos.adb:6:07: call to abstract function  
must be dispatching
```

```
gnatmake: "foos.adb" compilation error
```

What is the compiler trying to tell me here? And how do I go about calling abstract procedures?

[...]

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Fri, 20 Nov 2009 14:27:14 +0100

Subject: Re: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

[...]

It tells you that the type of THIS is FOO, so you cannot call to A, because A is not defined on FOO.

If P is to be defined in terms of any type from the class FOO, then P has to be declared differently (class-wide):

```
package FOOS is
  type FOO is abstract tagged
    null record;
  procedure P (THIS : in FOO'Class);
    -- I am SAME for the whole class
    -- rooted in FOO
  procedure A (THIS : in FOO) is
    abstract;
    -- I have an implementation in
    -- each instance of the class except
    -- for the abstract ones like FOO.
end FOOS;
package body FOOS is
  procedure P (THIS : in FOO'Class)
    is
  begin
    A (THIS);
    -- I do not know what kind of FOO is
    -- THIS. So I dispatch to A according
    -- to the tag of THIS.
  end P;
end FOOS;
```

From: Markus Schoepflin

<markus.schoepflin@comsoft.de>

Date: Fri, 20 Nov 2009 14:58:57 +0100

Subject: Re: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

> So If I understand this correctly, if I want dispatching to happen on a given type, I always need to use the class type?

And one more question, can I have a dispatching procedure that doesn't explicitly use the type it dispatches on? In C++, that would read:

```
class foo
{
  virtual void f() = 0;
};
class bar
{
  void f() {
```

```
  // Do something just depending on
  // class type bar,
  // not on an object of the class, so the
  // 'this' pointer actually is never used.
}
};
```

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Fri, 20 Nov 2009 15:10:05 +0100

Subject: Re: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

> So If I understand this correctly, if I want dispatching to happen on a given type, I always need to use the class type?

Exactly. This helps both the compiler to make the program much more efficient by eliminating unnecessary dispatch and you indicating the design problems like this and making it safer in the opposite cases:

Consider the case where one non-abstract operation calls another:

```
type FOO is abstract
  tagged null record;
procedure P (THIS : in out FOO);
procedure A (THIS : in out FOO);
...
procedure P (THIS : in FOO) is
  begin
    A (THIS);
    -- This is a part of the implementation
    -- of P for the type FOO
  ...
end P;
```

Here the call to A does not dispatch. In other language like C++ it would.

Now consider a type derived from FOO, that overrides A, but inherits P.

That could break the implementation of P in C++, but not in Ada, where P will behave exactly as it did before.

When you design P you should decide whether it is a class-wide subprogram or not. That role of P will define the behavior of calls to A from its body. Sometimes the compiler can detect that the declared role of P does not match its implementation, as it was in your case.

From: Niklas Holsti

<niklas.holsti@tidorum.fi>

Date: Fri, 20 Nov 2009 15:56:50 +0200

Subject: Re: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

[...]

```
> package body FOOS is
>   procedure P (THIS : in FOO)
>   is
>   begin
```

> A (THIS);

At this point, the compiler knows that THIS is a FOO object, or is to be seen as a FOO object. But you have said that the procedure A on FOO is abstract -- not implemented -- so you cannot call it.

I assume that your intention is to call the procedure A that is implemented (overridden) for the actual object THIS, which is of some type derived from FOO .. some type in FOO'Class for which you have implemented A. To do so, you must ask the compiler to make this call dispatching, by converting the parameter to FOO'Class:

A (FOO'Class (THIS));

This is called a "redispaching" call, because the procedure P may have been reached as the result of a dispatching call on P, and now we are asking to dispatch again.

One way to understand this is that in Ada by default calls are statically bound, not run-time bound (dispatching). Only calls that have parameters of class type are dispatching.

[...]

From: Niklas Holsti

<niklas.holsti@tidorum.fi>

Date: Fri, 20 Nov 2009 16:00:08 +0200

Subject: Re: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

> [...] So If I understand this correctly, if I want dispatching to happen on a given type, I always need to use the class type?

Yes. You can do that either by Dmitry's method, declaring the parameter as FOO'Class, or by converting a FOO parameter to FOO'Class for the call. But note that if an operation has no parameters (and no return value) of type FOO (or access FOO), only of FOO'Class, it is not a primitive operation of FOO and cannot be overridden in derived types.

From: Niklas Holsti

<niklas.holsti@tidorum.fi>

Date: Fri, 20 Nov 2009 16:19:57 +0200

Subject: Re: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

> [...] And one more question, can I have a dispatching procedure that doesn't explicitly use the type it dispatches on?

No, as there is no implicit "this" parameter in Ada (and no syntactic brackets to group all the operations of a type). To make an operation overridable (a "primitive operation" in Ada terms) you have to include one or more parameters of the type (or of "access" to the type) or it must return a value of the type (or "access" to the type). And moreover the operation must be declared

in the same package declaration as the type.

*From: Peter C. Chapin
<pcc482719@gmail.com>*

Date: 21 Nov 2009 14:07:12 GMT

Subject: Re: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

[...]

The 'this' pointer might never be used, but it is still there anyway. Ada's approach to object oriented programming requires that you make 'this' explicit. You can, of course, choose to ignore it as well.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Date: Fri, 20 Nov 2009 06:54:01 -0800 PST

Subject: Re: What makes a procedure call 'dispatching' in Ada?

Newsgroups: comp.lang.ada

I'd like to follow up on the (correct) replies in this thread so far with a "dynamic dispatching in Ada for C++ programmers" primer.

In C++:

```
class C {
    virtual void foo ();
}
C object;
C* pointer = &object;
void p () {
    object.foo (); // static dispatch
    pointer->foo (); // dynamic dispatch
}
```

In C++, class types are specific and pointer types are class-wide. The declaration

```
C* pointer = &object;
```

is strictly equivalent to

```
Object : aliased C;
type C_Access is access all C'Class;
Pointer : C_Access := Object'Access;
```

Ada makes it explicit that Pointer is of a class-wide type, therefore calls through the pointer dispatch dynamically:

```
procedure P is
begin
    Foo (Object); -- static dispatch
    Foo (Pointer.all); -- dynamic dispatch
end P;
```

So far it seems that Ada and C++ are really the same, but wait! Ada has a syntax to declare access types to a specific type, like so:

```
type C_Specific_Access is
    access all C;
```

Any calls to primitive operations of C through C_Specific_Access will dispatch statically, not dynamically. There is no way in C++ to declare such a type. In

C++, all pointer types are class-wide; this applies also to the type of the implicit "this" parameter.

Conversely, C++ has no way to declare a class-wide type that is not a pointer or reference type. Ada has C'Class for just this purpose. The consequence is that, in Ada, you do not need any pointers to achieve dynamic dispatching whereas C++ requires you to use pointers if you want dynamic dispatching. Consider again:

```
void p () {
    object.foo (); // static dispatch
    pointer->foo (); // dynamic dispatch
}
```

There is no way to dispatch dynamically on "object"; you must use "pointer"; contrast with Ada:

```
procedure P (Object : C) is
begin
    Foo (Object); -- static dispatch
    Foo (C'Class (Object)); -- safe
    -- dynamic dispatch, without pointers!
end P;
```

The construct C'Class (Object) is called a "view conversion" in Ada; it entails no run-time cost and no additional object code in this case (convert "up" the type hierarchy) but it allows the programmer to choose whether each call should dispatch statically or dynamically.

Bug in the implementation of Timing Events

From: Reto Buerki <reet@codelabs.ch>

Date: Mon, 14 Sep 2009 18:12:14 +0200

Subject: Timing_Events: Event time still set after Cancel_Handler

Newsgroups: comp.lang.ada

[...]

Before reporting a bug, I wanted to ask your opinion on this. Consider the following code:

```
with Ada.Real_Time.Timing_Events;
package Timers is
    use Ada.Real_Time.Timing_Events;
    protected type Timer_Type is
        procedure Setup (
            At_Time : Ada.Real_Time.Time);
        function Get_Time return
            Ada.Real_Time.Time;
        procedure Stop (
            Status : out Boolean);
    private
        procedure Handle (
            Event : in out Timing_Event);
    end Timer_Type;
end Timers;
--
package body Timers is
```

```
protected body Timer_Type is
    function Get_Time return
        Ada.Real_Time.Time is
    begin
        return Event.Time_Of_Event;
    end Get_Time;
    procedure Handle (
        Event : in out Timing_Event) is
    begin
        null;
    end Handle;
    procedure Setup (
        At_Time : Ada.Real_Time.Time) is
    begin
        Event.Set_Handler (
            At_Time => At_Time,
            Handler => Handle'Access);
    end Setup;
    procedure Stop (
        Status : out Boolean) is
    begin
        Event.Cancel_Handler (
            Cancelled => Status);
    end Stop;
    end Timer_Type;
end Timers;
--
with Ada.Text_IO;
with Ada.Real_Time;
with Timers;
procedure Cancel_Handler is
    use Ada.Real_Time;
    Handler : Timers.Timer_Type;
    Timer : constant Time :=
        Clock + Minutes (60);
begin
    if Handler.Get_Time = Time_First
    then
        Ada.Text_IO.Put_Line (
            "Time is Time_First ...");
    end if;
    Handler.Setup (At_Time => Timer);
    if Handler.Get_Time = Timer then
        Ada.Text_IO.Put_Line (
            "Handler set ...");
    end if;
    declare
        Stopped : Boolean := False;
    begin
        Handler.Stop (Status => Stopped);
        if Stopped then
            Ada.Text_IO.Put_Line (
                "Timer cancelled ...");
            if Handler.Get_Time = Timer then
                Ada.Text_IO.Put_Line (
                    "Why is the time still set then?");
            end if;
        end if;
    end;
end;
end Cancel_Handler;
```

The 'Timers' package provides a simple protected type 'Timer_Type' which basically just wraps an Ada.Real_Time.Timing_Events.Timing_Event type.

The Setup() procedure can be used to set a specific event time.

Stop() just calls the Cancel_Handler() procedure of the internal Timing_Event. This procedure 'clears' the event (if it is set).

Ada RM D.15 about Real_Time.Timing_Events states:

9/2 An object of type Timing_Event is said to be set if it is associated with a non-null value of type Timing_Event_Handler and cleared otherwise. All Timing_Event objects are initially cleared.

17/2 The procedure Cancel_Handler clears the event if it is set. Cancelled is assigned True if the event was set prior to it being cleared; otherwise it is assigned False.

18/2 The function Time_Of_Event returns the time of the event if the event is set; otherwise it returns Real_Time.Time_First.

The RM does not explicitly state what happens with the event time value associated with a specific event after a call to Cancel_Handler(), but it seems logical to assume that Time_Of_Event should return Time_First again because no event is 'set' after it has been cleared (handler set to 'null').

With FSF GNAT, only the event handler is cleared, the event time remains set. Tested with GNAT 4.3.2 and 4.4.1 on Debian Stable/SID.

Could this be considered as a bug?

[...]

From: Reto Buerki <reet@codelabs.ch>
Date: Wed, 16 Sep 2009 23:19:51 +0200
Subject: Re: Timing_Events: Event time still set after Cancel_Handler
Newsgroups: comp.lang.ada

[...]

> I say it is a bug because ARM 18/2 is violated.

Thanks for the confirmation, bug filed [1].

[...]

[1] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=41383

On the Ravenscar Profile and ACATS

From: Jérôme Hugues <hugues@telecom-paristech.fr>
Date: Fri, Oct 2 2009 16:03:00 CEST
Subject: [ada-france] Question sur le profil Ravenscar
Mailing list: ada-france.ada-france.org

I take advantage of the change of the Board to take the role of the inquisitor;-)

In Ada 2005 the Ravenscar Profile was officially defined. At ada-auth.org it is possible to read the minutes of the discussions on the genesis of the profile.

While reading it again, I found the following

(<http://www.ada-auth.org/cgi-bin/cvsweb.cgi/ais/ai-00249.txt?rev=1.16>):
!ACATS test

An ACATS test should be created for this pragma.

Does it mean there is no test?

As a subordinate question, is there a compiler fulfilling the ACATS tests for the Ada 95 Annex D on Real-Time Systems, so that the aspects related to Task_Dispatching_Policy and Locking_Policy are validated, as well as the behavioural part of Ravenscar, with the exception of the pragma Detect_Blocking?

The problem I have is to understand if RTEMS is compliant to Ravenscar or if we risk to find a problem in the future (under the hypothesis that it passes the ACATS tests).

From: Laurent Guerby <laurent@guerby.net>
Date: Fri, Oct 2 2009 16:20:00 CEST
Subject: Re: [ada-france] Question sur le profil Ravenscar
Mailing list: ada-france.ada-france.org

[...]

I did not find "Ravenscar" in the last updated ACATS tests:

<http://www.ada-auth.org/acats.html>
[...]

ACATS tests of Annex D are not included in the test suite of GCC, someone has to test them (just copy the files of the cxd folder of ACATS in gcc/testsuite/ada/ACATS/tests/cxd/).

From: Jérôme Hugues <hugues@telecom-paristech.fr>
Date: Fri, Oct 2 2009 16:35:00 CEST
Subject: Re: [ada-france] Question sur le profil Ravenscar
Mailing list: ada-france.ada-france.org

[...]

Thanks for that link, I forgot it!

[...]

Mmm, do you know where are the scripts to run the suite to RTEMS?

(/trunk/gcc/testsuite/ada/acats/run_all.sh seems only to run the test case on native)

I imagine that you can easily patch them to specifically test this directory and RTEMS with a BSP for LEON3, am I wrong?

From: Laurent Guerby <laurent@guerby.net>
Date: Fri, Oct 2 2009 17:31:00 CEST

Subject: Re: [ada-france] Question sur le profil Ravenscar

Mailing list: ada-france.ada-france.org

Joel Sherrill has the patch to do that, he sends me the results and issues from time to time but it has been a while since I performed a cross test on RTEMS.

In run_all.sh it is sufficient to change the 4 small functions target_run/gnatchop/gnatmake/gcc to adapt them to your target.

The easiest way if you have any problem is to contact Joel directly, you can tell him I told you to contact him :).

[The whole thread was translated from French —mp]

Use of the same actual parameter for in and out formal parameter

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 17 Nov 2009 01:50:04 -0800 PST
Subject: Passing the same actual as both in and out formal parameters?
Newsgroups: comp.lang.ada

Consider:

```
type T is tagged private;
procedure P (A : in T; B : out T)
is separate;
Object : T;
begin
  P (A => Object, B => Object);
```

This seems legal but I suspect the execution might lead to bugs if P reads and writes components of A and B in arbitrary order, e.g.

```
type T is tagged record
  L, M : Integer;
end record;
procedure P (A : in T; B : out T) is
begin
  B.L := A.M; -- does this change
             -- A.L too?
  B.M := A.L; -- bug: A.L has been
             -- clobbered, now B.M = B.L?
end P;
```

My concern stems from the fact that T is tagged (I cannot change that), so Object is passed by reference as both A and B.

Am I right to be concerned?

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Tue, 17 Nov 2009 12:40:57 +0200
Subject: Re: Passing the same actual as both in and out formal parameters?
Newsgroups: comp.lang.ada

[...]

> This seems legal but I suspect the execution might lead to bugs if P reads

and writes components of A and B in arbitrary order, e.g.

I think this situation is defined in RM 6.2(12) where A and B are defined as "distinct access paths" to the same object. It is a bounded error if the parameter passing mechanism is not specified, but (by default) should work as expected when the parameters are passed by reference.

> type T is tagged record

> L, M : Integer;

> end record;

>

> procedure P (A : in T; B : out T) is

> begin

> B.L := A.M; -- does this change A.L too?

Yes, as far as I understand RM 6.2(12).

> B.M := A.L; -- bug: A.L has been clobbered, now B.M = B.L?

I believe so.

[...]

> My concern stems from the fact that T is tagged (I cannot change that), so Object is passed by reference as both A and B.

> Am I right to be concerned?

Yes, if you expect A to be immutable during the execution of P.

There is a Note to RM 6.2(12), which is 6.2(13): "A formal parameter of mode in is a constant view (see 3.3); it cannot be updated within the subprogram body". But I think this means only that the "in" mode access path to this object cannot be used to update it. It does not mean that the value of the object cannot change at all, due to assignments from other access paths.

If P is meant to return B as A with L and M swapped, you should use an aggregate assignment, B := (L => A.M, M => A.L).

From: Jean-Pierre Rosen
<rosen@adalog.fr>

Date: Tue, 17 Nov 2009 11:31:02 +0100

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

[...]

> Am I right to be concerned?

Depend on what your concern is ;-)

The semantic is well defined: tagged types are by-reference type. If you want to swap two fields of different parameters of the same type, use a local variable.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Date: Tue, 17 Nov 2009 03:26:29 -0800

PST

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

[...]

My example was heavily simplified; the actual type has about a hundred components and the procedure P is a little more complex than swapping components :)

But thanks for the responses, Niklas and Jean-Pierre. They confirm my suspicion.

From: Jean-Pierre Rosen

<rosen@adalog.fr>

Date: Tue, 17 Nov 2009 14:13:25 +0100

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

[...]

But the important thing is that there is no risk: behaviour is well defined, and will not change with the next release of the compiler.

If it is not the behaviour you want, you can make a local copy (but you know that).

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Date: Tue, 17 Nov 2009 08:07:39 -0800

PST

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

[...]

Indeed; I was careful not to use the phrases "bounded error" or "erroneous execution", just "bug" :) The construct is well-defined but error-prone and needs documentation in my sources, so I've added that.

Triggering the bug requires:

(1) pass-by-reference type (i.e. tagged, limited, etc.) or explicit access type

(2) same object passed twice as both in and out parameters

(3) non-atomic reads and writes to the object inside the subprogram

When writing such a procedure, it is necessary to pay attention: either prevent the bug by checking for condition (2) and raising an exception if it is met; use only atomic operations so as to prevent (3); or accept that the bug may happen and warn about it.

In my particular case, the operations are "atomic" in that the procedure first reads the Object, then passes it as an "out" parameter to another procedure, and never reads it again. However, this being long-term-support software, one never knows that (3) can never happen in some future revision.

From: Adam Benesch

<adam@irvine.com>

Date: Tue, 17 Nov 2009 08:26:47 -0800

PST

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

[...]

> Am I right to be concerned?

As the others have pointed out, the answers to your questions are "yes", changing B.L does change A.L if the same object is passed as a parameter to both A and B. The semantics are well-defined. My concern would be whether optimization could change the order of the operations inside P in a way that affects the results if A and B are aliases for the same object; I don't know offhand whether this is allowable for parameters of by-reference types. I'd have to hunt through the RM to figure this out, unless someone already knows the answer.

Whether this (the simpler problem, without optimization) is a concern or not depends on the situation. I've written procedures that are specifically designed to allow the same object to be passed as an IN and an OUT parameter.

Of course, the body of the procedure has to be written carefully to allow for this. There's no way in Ada to enforce any of this; right now it's just mentioned in the comments in the package spec ("A and B may be the same object", or "A and B may not be the same object"), and the caller is expected to obey this, and the body is expected to perform correctly when they are the same object, if they are indeed allowed to be the same.

I think AI05-191 is related to this. Offhand, it appears that if this AI is addressed, you could put an assertion somewhere (as a precondition of P, if AI05-145 is addressed) to ensure that P is never called with aliased (or overlapping) components, if that would be bad.

From: Randy Brukardt

<randy@rrsoftware.com>

Date: Tue, 17 Nov 2009 15:25:13 -0600

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

> My concern would be whether optimization could change the order of the operations inside P in a way that affects the results if A and B are aliases for the same object; [...]

I don't believe such an optimization would be legitimate, but given how hard it is to understand 11.6, I could be wrong. Most of the permissions to optimize are related to whether (and where) exceptions are raised, but there is no exception here. Similarly, there are a lot of things that are evaluated in an unspecified order (which could change because of optimization), but that also does not apply here.

> I think AI05-191 is related to this. Offhand, it appears that if this AI is addressed, you could put an assertion somewhere (as a precondition of P, if AI05-145 is addressed) to ensure that P is never called with aliased (or

overlapping) components, if that would be bad.

Right, that's my understanding of the point. The only problem is, there isn't any sane way to describe such an assertion.

[...]

P.S. The checks proposed in AI05-0144-1 also are related to this situation, although they would not detect this particular case (as the semantics is well-defined, there is no non-portability here).

From: Jeffrey R. Carter

<jrcarter@acm.org>

Date: Tue, 17 Nov 2009 17:11:26 -0700

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

> [...] The only problem is, there isn't any sane way to describe such an assertion.

Given that the types are by-reference, would comparing 'access of the parameters serve?

pragma Assert (A'access /= B'access);

From: Adam Benesch

<adam@irvine.com>

Date: Tue, 17 Nov 2009 16:23:56 -0800

PST

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

[...]

First of all, for this to work in the general case, that would need a major change in language semantics, since you need an access type in order for 'Access to be allowed. The only way this would be legal is if there happened to be exactly one "=" operator directly visible with operands of some named access-to-T type. (Also, if "=" were overridden with a user-defined operator that did something unexpected, it would fail, but nobody would do that.)

Second, it only catches the case where the operands are of the same type; it won't catch other overlaps such as

```
P2 ( A => Object,
      B => Object.Component);
```

From: Jeffrey R. Carter

<jrcarter@acm.org>

Date: Tue, 17 Nov 2009 20:47:53 -0700

Subject: Re: Passing the same actual as both in and out formal parameters?

Newsgroups: comp.lang.ada

[...]

OK. Since this in the body of the operation, it seems doable:

```
procedure P ( A : in T; B : out T) is
  type T_Ptr is access T;
  A_Ptr : constant T_Ptr := A'access;
  B_Ptr : constant T_Ptr := B'access;
  pragma Assert (A_Ptr /= B_Ptr);
```

So you're guaranteed that "=" for T_Ptr is used, and you know it hasn't been

overridden.

What about 'Address?

> Second, it only catches the case where the operands are of the same type; it won't catch other overlaps such as

```
> P2 ( A => Object, B =>
      Object.Component);
```

Sure, it's not a general solution; I doubt if there could be one. But it does seem to serve for the OP's case.

On tagged type and access to subprograms

From: Yannick Duchêne

<yannick_duchene@yahoo.fr>

Date: Wed, 28 Oct 2009 00:51:19 -0700

PDT

Subject: Tagged type more type safe than access to subprogram ?

Newsgroups: comp.lang.ada

[...]

I was reading the Ada 95 Quality and Style Guide, seeking for some inspiration about a design / style doubt.

I came into Chapter 5, "CHAPTER 5: Programming Practices",

5.3.4 Subprogram Access Types says:

> You can achieve the same effect as access-to-subprogram types for dynamic selection by using abstract tagged types. You declare an abstract type with one abstract operation and then use an access-to-class-wide type to get the dispatching effect. This technique provides greater flexibility and type safety than access-to-subprogram types

Here :

http://www.iste.uni-stuttgart.de/ps/ada-doc/style_guide/sec_5a.html#5.3.4

I agree about the "greater flexibility" (I've recently meet such a case), but I do not understand the "and [greater] type safety".

If it's Ok for me to assert that tagged type is a more flexible way than access to subprogram, I do not see a case where access to subprogram would be less type safe than tagged type.

If there is something I do not understand, this may mean I have something to learn about it (the purpose of the question then).

Does any one know a case which match this assertion ?

[...]

From: Dmitry A. Kazakov

<mailbox@dmitry-kazakov.de>

Date: Wed, 28 Oct 2009 09:55:08 +0100

Subject: Re: Tagged type more type safe than access to subprogram ?

Newsgroups: comp.lang.ada

[...]

An access to subprogram is a poor-man's closure. Let's ignore "access" part and

consider a pure downward closure (as it should have been in Ada).

I argue that a helper type with an abstract primitive subprogram is safer than a closure, both safer and type-safer.

The general safety comes from the fact that a closure brings a context with it, which an object normally does not. An abstract type is better encapsulated and there is less chances to run into occasional side effects.

Even if the side effects are desired I would argue that it is better and safer to limit them to the object rather than to the closure's context. The crucial point is that closure's effects are "there", while effects on the object are localized to the object's state, there are "here", at the call point.

The type safety is gained through different types derived from the abstract parent. There might be no difference at the caller side, e.g. whether you mistakenly call to the closure C1 instead of C2, or mistakenly pass an instance of S1 instead of S2 (both from the S'Class). Yet it is type safer with regard to the parameters required to construct S2. Its constructor can have a different signature, so that the parameters you pass in order to create it were different from S1.

From: Yannick Duchêne

<yannick_duchene@yahoo.fr>

Date: Thu, 29 Oct 2009 04:00:29 -0700

PDT

Subject: Re: Tagged type more type safe than access to subprogram ?

Newsgroups: comp.lang.ada

[...]

> An access to subprogram is a poor-man's closure. Let's ignore "access" part and consider a pure downward closure (as it should have been in Ada).

"As this should have been in Ada" ? What were you meaning? I've always though real closures are not possible with such structures as Ada provides, except at package level — which is especially the case when a package can have multiple instances... but only at package level. Isn't it ?

> I argue that a helper type with an abstract primitive subprogram is safer than a closure, both safer and type-safer. [...]

> The type safety is gained through different types derived from the abstract parent. [...]

If I attempt an abstract of your words, and if I've understood you in a right way, it could be: tagged types are safer than accesses to subprograms, because the abstract method of a tagged type is always associated to a suitable context, unlike subprogram, with which a single error is immediately turned into a double error — the one about the subprogram and the one about the closure which comes with the

subprogram reference.

This is relevant, indeed.

Now I see the deal much better (providing I'm OK with my understanding).

I would like to just add two other notes, about flexibility (not safety): the first one, I think its a good idea to add a Ready function (returning Boolean) to the tagged type, because the tagged type may be extended in a way where it needs setup (useful for peoples who like to add Eiffel-like precondition to their specifications).

This seems general enough to me to add this function to any tagged type which is intended to act as an abstract method (and to add an initial default implementation which always return True). And then, about implementation now : if an abstract method is concerned by performance, it is possible to add an alternate batch version belong to the primary abstract method (to run the method on a sequence of parameters, either as input or output). This is a good idea to add it to the same tagged type, as it is deeply related to the "normal" (un-batched version) of the abstract method (deeply related in many ways, such as required internal data to work, setup, shared part of algorithm, etc).

This is another argument pleading in favor of the provided flexible of tagged types over access to subprograms.

*From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Thu, 29 Oct 2009 18:54:19 +0100
Subject: Re: Tagged type more type safe than access to subprogram ?
Newsgroups: comp.lang.ada
[...]*

>> An access to subprogram is a poor-man's closure. Let's ignore "access" part and consider a pure downward closure (as it should have been in Ada).

[...]

I meant downward closures. There is no reason for

type P is access procedure (...);

where actually meant

type P is procedure (...);

It is almost always safer and cleaner to pass subprograms instead of access to them. Obviously a subprogram type were a limited type, so a subprogram were passed by reference. A pointer were only needed when you wanted to copy it. And copying pointers is always asking for trouble...

BTW, in Ada 83, there was no access to subprogram, so we used tasks instead (where a subprogram had to be a non-generic parameter).

Task is a proper type since the day one. Subprograms lingered, but then in Ada 95 one did a big mistake introducing access

discriminants, access to subprogram, access to self (the Rosen trick), access to function's mutable parameter etc. And almost in all use cases of these, no access is actually needed.

*From: Georg Bauhaus
Date: Thu, 29 Oct 2009 21:45:02 +0100
Subject: Re: Tagged type more type safe than access to subprogram ?
Newsgroups: comp.lang.ada*

> BTW, in Ada 83, there was no access to subprogram, so we used tasks instead (where a subprogram had to be a non-generic parameter).

Tagged types may come close to a solution sometimes. You make "function objects", like the ones found in Eiffel and, I think, some other languages. [...]

*From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Fri, 30 Oct 2009 09:25:12 +0100
Subject: Re: Tagged type more type safe than access to subprogram ?
Newsgroups: comp.lang.ada
[...]*

> Tagged types may come close to a solution sometimes.

Yes, I am using this pattern very often. But:

1. There were no tagged types in Ada 83
2. In Ada 95 tagged types were in effect strictly library-level. This restriction was lifted only in Ada 2005, too late to stop the "access-to everything cancer".
3. The language is too heavy when it comes to create a singleton object overriding one or two abstract primitive operations. There should be short-cuts for that.

Null range in unconstrained array

*From: Rick Duley <rickduley@gmail.com>
Date: Mon, 31 Aug 2009 18:28:06 -0700 PDT
Subject: Null Range in Unconstrained Array
Newsgroups: comp.lang.ada*

If I have an unconstrained array type 'My_Array_Type' and declare an instance of it as:

```
My_Array : My_Array_Type (1 .. 0);
```

then the LRM tells me it is a null range:

3.5 (4): A range with lower bound L and upper bound R is described by "L .. R".

If R is less than L, then the range is a null range, and specifies an empty set of values.

What, exactly, is My_Array (forgive the language) pointing to? Is any memory allocated to My_Array?

*From: Randy Brukardt
<randy@rrsoftware.com>*

*Date: Mon, 31 Aug 2009 22:11:23 -0500
Subject: Re: Null Range in Unconstrained Array
Newsgroups: comp.lang.ada
[...]*

That's a question that the ARG decided not to answer in general. (It matters for aliased objects and "=" of accesses to such objects). We agreed to not decide (see AI95-00350-1, voted No Action). It seems silly to require allocating memory for objects with no components, but some people think that it is important that access type compare unequal. Thus there was no agreement on clarifying the wording.

So you'll have to look to see what your particular compiler does.

*From: Adam Benesch
<adam@irvine.com>
Date: Tue, 1 Sep 2009 07:50:48 -0700 PDT
Subject: Re: Null Range in Unconstrained Array
Newsgroups: comp.lang.ada
[...]*

Most likely, no. Randy gave a reason why a compiler might want to allocate a little bit of space for the array; but even if it does, it's memory that will never be used. Any attempt to refer to My_Array (X), no matter what X is, will raise Constraint_Error.

*From: Robert A Duff
<bobduff@shell01.TheWorld.com>
Date: Tue, 01 Sep 2009 11:34:30 -0400
Subject: Re: Null Range in Unconstrained Array
Newsgroups: comp.lang.ada
[...]*

Well, arrays don't "point".

>> Is any memory allocated to My_Array?

> Most likely, no. Randy gave a reason why a compiler might want to allocate a little bit of space for the array; but even if it does, it's memory that will never be used.

Randy's reason applies only if the array is aliased, which the above one is not.

There's no reason the compiler has to allocate any space for My_Array above.

If we have:

```
My_Array : My_Array_Type (1 .. 0);  
X : Boolean;
```

it is entirely possible (likely even) that My_Array'Address = X'Address.

Randy's concern about access values does not apply to addresses.

Usually, empty arrays are not statically known to be empty, though. In that case, some space might be used to store the bounds.

>...Any attempt to refer to My_Array (X), no matter what X is, will raise

Constraint_Error.

Right.

From: Peter C. Chapin
<pcc482719@gmail.com>

Date: 06 Sep 2009 12:11:05 GMT

Subject: Re: Null Range in Unconstrained
Array

Newsgroups: comp.lang.ada

[...]

Wouldn't it be possible to pass My_Array to a subprogram expecting My_Array_Type? The subprogram might consult the bounds on the formal parameter before trying to use the array so sending a null array to such a subprogram is not automatically going to cause a problem. In that case, it seems like My_Array needs to have memory allocated for it to hold information about the bounds.

From: Robert A Duff
<bobduff@shell01.TheWorld.com>

Date: Sun, 06 Sep 2009 08:41:22 -0400

Subject: Re: Null Range in Unconstrained
Array

Newsgroups: comp.lang.ada

> Wouldn't it be possible to pass
My_Array to a subprogram expecting
My_Array_Type?

Yes.

>...The subprogram might consult the
bounds on the formal parameter before
trying to use the array [...]

The called procedure has to have some
way to know the bounds.

That's true whether the bounds are 1..0 or
1..100.

There are several ways to implement that.

For example, the procedure could be
passed the bounds as separate parameters,
in two registers. I don't consider that to be
"memory allocated for My_Array",
because it is not allocated when the
compiler sees My_Array -- it is allocated
when the compiler sees a call (and
separately for each call).

In this case, My_Array'Address =
X'Address is likely.

Alternatively, the compiler could allocate
the bounds as part of My_Array, just in
case there are some such calls.

In this case, My_Array'Address =
X'Address is unlikely.

If the procedure is inlined, the bounds
might end up being stored only in the
immediate-value fields of instructions. Or
the entire procedure call might vanish,
because the compiler knows it's (say)
looping through an empty array.

From: Adam Benesch
<adam@irvine.com>

Date: Tue, 8 Sep 2009 10:54:08 -0700 PDT

Subject: Re: Null Range in Unconstrained
Array

Newsgroups: comp.lang.ada

[...]

To elaborate on this a bit further: Suppose
you define a record type for reading a file
with Ada.Direct_IO, that looks something
like this:

type Employee_Data **is record**

 Name : String (1 .. 50);

 Address1 : String (1 .. 40);

 Address2 : String (1 .. 40);

 City : String (1 .. 30);

 State : String (1 .. 2);

 ...

end record;

[...] it would be very unexpected for the
compiler to put two extra integers in the
record for each of these strings, and it
would mess up your file I/O. But you
have to have the ability to pass any of
those fields to a subprogram with a
parameter type "String", and the bounds
have to be passed to the subprogram
somehow. I think it's most likely that the
bounds will be passed separately, as Bob
suggested, either by passing them in
separate registers, creating a three-word
temporary structure that contains the
bounds and a pointer to the data and
passing the address of that structure (with
that structure disappearing after the
subprogram returns), creating a two-word
structure with the bounds and passing the
address of that in a register, etc.
Something along those lines. But I'd
actually be surprised if *any*
implementation allocated extra space in
an Employee_Data record to hold the
bounds of each field.

From: Stephen Leake

<stephen_leake@stephe-leake.org>

Date: Wed, 09 Sep 2009 04:35:19 -0400

Subject: Re: Null Range in Unconstrained
Array

Newsgroups: comp.lang.ada

> [...] it would be very unexpected for the
compiler to put two extra integers in the
record for each of these strings, and it
would mess up your file I/O.

If you expect the layout of this record to
match your file, you must provide a
representation clause.

> But you have to have the ability to pass
any of those fields to a subprogram
with a parameter type "String", and the
bounds have to be passed to the
subprogram somehow. [...]

It could depend on whether there's a
pragma Pack, or a representation clause,
for the record. In the absence of such, I
would expect the compiler to treat these
components the same way it treats
separate objects. Which means I would
expect the bounds to be stored with the
object.

But that's just my expectations; compilers
are free to do whatever they want, as long
as it meets the standard.

From: Robert A Duff

<bobduff@shell01.TheWorld.com>

Date: Wed, 09 Sep 2009 09:00:39 -0400

Subject: Re: Null Range in Unconstrained
Array

Newsgroups: comp.lang.ada

[...]

>... Which means I would expect the
bounds to be stored with the object.

Most compilers, including GNAT, will
not store the bounds with the object
(whether it's a component or a standalone
object).

Think about an array of a million of those
records: you don't want 1,000,000 copies
of the number 50 stored at run time. That
number is known to the compiler, and can
be plugged in wherever needed (e.g. when
you say "for X in A(Y).Name'Range
loop", the compiler knows that's just "for
X in 1..50 loop".

> But that's just my expectations;
compilers are free to do whatever they
want, as long as it meets the standard.

Right.

From: Simon J. Wright

<simon.j.wright@mac.com>

Date: Wed, 9 Sep 2009 12:22:42 -0700 PDT

Subject: Re: Null Range in Unconstrained
Array

Newsgroups: comp.lang.ada

[...]

> type Employee_Data is record

> [...]

> end record;

After leaving out the ..., GNAT allocates
162 bytes for this record (no dope bytes).
Also, if you stream it, it takes just 162
bytes on the stream.

From: Stephen Leake

<stephen_leake@stephe-leake.org>

Date: Thu, 10 Sep 2009 19:24:40 -0400

Subject: Re: Null Range in Unconstrained
Array

Newsgroups: comp.lang.ada

> Also, if you stream it, it takes just 162
bytes on the stream.

That's not surprising: LRM 13.13.2 says:

3 S'Write

S'Write denotes a procedure with the
following specification:

4/2 **procedure** S'Write(
 Stream : **not null access**

 Ada.Streams.Root_Stream_
 Type'Class;
 Item : in T)

5 S'Write writes the value of Item to
Stream.

The value clearly does not include the
bounds.

Binary I/O on standard input/output stream

From: Yannick Duchêne

<yannick_duchene@yahoo.fr>

Date: Sat, 24 Oct 2009 15:07:29 -0700 PDT

Subject: Preferred way to do binray I/O on standard input/output stream

Newsgroups: comp.lang.ada

[...]

There is no kind of binary mode vs text mode flags with Ada stream/file IO, as it there are with C. And then, C files are not typed, while Ada files/streams are, thus, this will not make any sense to simply talk about binary I/O with Ada. So let simply talk about non-text I/O from and to standard input/output.

When some people need this, what is the typical preferred to do so ?

Using a custom file type which relies on the OS ? (one for each OS if it is to be built for multiple platforms)

Simply use characters with Text_IO and then do a character code interpretation as if it was binary data? Other ways?

The first way, seems the more formal, but requires more work, and probably a lot of testing to ensure the custom file type implementation does not contain any error.

The second way seems to be safer in some way (relies on a standard package), but does not seem safe in the way nothing can ensure no data will not be lost or modified (as it is primarily Text_IO).

If I missed something in the standard packages which allow this to be done directly, I simply apologize for this topic.

[...]

From: Jeffrey R. Carter

<jrcarter@acm.org>

Date: Sat, 24 Oct 2009 15:57:25 -0700

Subject: Re: Preferred way to do binray I/O on standard input/output stream

Newsgroups: comp.lang.ada

[...]

Package Ada.Text_IO.Text_Streams (ARM A.12.2) allows converting Ada.Text_IO.File_Type to Stream_Access and reading and writing them through the capabilities of streams. This allows converting standard input and output to Stream_Access, since Ada.Text_IO provides the functions Standard_Input and Standard_Output which return Ada.Text_IO.File_Type.

For general binary I/O, one can use Ada.Sequential_IO or Ada.Direct_IO instantiated with an appropriate type, or Ada.Streams.Stream_IO. For standard input and output, however, this can only work if you have a name for these files that you can use to open them, and such a name is platform-dependent. The only independent way to access these files is

through streams and
Ada.Text_IO.Text_Streams.

From: Yannick Duchêne

<yannick_duchene@yahoo.fr>

Date: Sat, 24 Oct 2009 16:22:41 -0700 PDT

Subject: Re: Preferred way to do binray I/O on standard input/output stream

Newsgroups: comp.lang.ada

[...]

There is further more a note in ARM 12.2, which says (annotated reference) :

NOTES

6 35 The ability to obtain a stream for a text file allows Current_Input, Current_Output, and Current_Error to be processed with the functionality of streams, including the mixing of text and binary input-output, and the mixing of binary input-output for different types.

7 36 Performing operations on the stream associated with a text file does not affect the column, line, or page counts.

[...]

From: Yannick Duchêne

<yannick_duchene@yahoo.fr>

Date: Mon, 26 Oct 2009 17:34:08 -0700

PDT

Subject: Re: Preferred way to do binray I/O on standard input/output stream

Newsgroups: comp.lang.ada

[...]

While this seems to work most of times, I meet a malfunction (data not preserved) if the stream ends with a line-feed (a byte whose value is 10). When it is, this last byte seems to be dropped from the stream, which cause troubles if the binary format is expecting it. This occurs only when it exactly ends with this byte, and it is Ok if this byte is followed by a byte whose value is 0 (as an example). It does not occur when the stream ends with a byte whose value is that of the carriage-return (13).

Finally, this does not really turn the stream into a binary stream, and some interpretations remain.

From: John B. Matthews

<jmatthews@wright.edu>

Date: Mon, 26 Oct 2009 21:14:38 -0400

Subject: Re: Preferred way to do binray I/O on standard input/output stream

Newsgroups: comp.lang.ada

[...]

I get the same effect. I don't know if it's the shell (bash) or OS (Mac) doing it. I suspect the former: End_Of_File turns True when a final linefeed remains to be read; the effect is absent with redirection. I'm vaguely uneasy using an exception for flow control, but this seems to copy the data unmolested:

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Text_IO.Text_Streams;
procedure Copy is
```

```
Stream_Ptr :
  Text_Streams.Stream_Access;
```

```
C : Character;
```

```
begin
```

```
Stream_Ptr := Text_Streams.Stream(
  Current_Input);
```

```
loop
```

```
  Character'Read(Stream_Ptr, C);
```

```
  Put(C);
```

```
end loop;
```

```
exception
```

```
when End_Error => null;
```

```
end Copy;
```

From: Yannick Duchêne

<yannick_duchene@yahoo.fr>

Date: Mon, 26 Oct 2009 19:36:49 -0700

PDT

Subject: Re: Preferred way to do binray I/O on standard input/output stream

Newsgroups: comp.lang.ada

[...]

The trouble with this, is that this force a look-ahead: an item must be read to know if an item is available, and this can lead into numerous logical traps (I prefer to keep distinct the action of reading and testing availability of data).

May be this is finally really better to re-create a type to stand for the standard input as binary, but what I do not like with this way, is the possible lack of knowledge of some platforms, which is required for implementations (For me, it will be OK for Windows, BSD, Linux, but not the others... although in the mean time, I'm not sure I will ever need it for other platforms).

From: John B. Matthews

<jmatthews@wright.edu>

Date: Tue, 27 Oct 2009 12:13:54 -0400

Subject: Re: Preferred way to do binray I/O on standard input/output stream

Newsgroups: comp.lang.ada

[...]

Ah, I see your point. I recall this problem going back to the days of UCSD Pascal, which distinguished between interactive and text type files for this very reason. My use is to allow command line utilities to read from standard input if no file name is supplied. For me, the data stream invariably arrives via redirection or a pipe, so the problem does not arise. For interactive programming, I typically use GtkAda.

This variation of Copy makes it easier to see what's happening.

Prefacing the loop with the End_Of_File predicate exposes the problem for files with a terminal LF. If running interactively, control-D exits:

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Text_IO.Text_Streams;
procedure Copy is
```

```

Stream_Ptr :
  Text_Streams.Stream_Access;
C : Character;
function Hex(C : Character)
  return String is
  H : constant String :=
    "0123456789ABCDEF";
  B : Natural := Character'Pos(C);
  S : String(1 .. 4);
begin
  S(1) := '[';
  S(2) := H(B / 16 + 1);
  S(3) := H(B mod 16 + 1);
  S(4) := ']';
  return S;
end Hex;
begin
  Stream_Ptr := Text_Streams.Stream(
    Current_Input);
-- while not End_Of_File loop
  loop
    Character'Read(Stream_Ptr, C);
    Put(Hex(C));
  end loop;
exception
  when End_Error => null;
end Copy;

```

[...]

On the performance of 'Read and 'Write

*From: Gautier de Montmollin
<gdemont@users.sourceforge.net>
Date: Thu, 29 Oct 2009 16:29:47 -0700 PDT
Subject: Performance of the Streams 'Read and 'Write
Newsgroups: comp.lang.ada*

[...]

I got used to think that I/O was the last spot where our preferred language was condemned to slowness.

Now consider this. Variant 1 of a buffered I/O:

```

type Buffer is array(Natural range <>)
  of Unsigned_8;
procedure Read( b: out Buffer ) is
begin
  Buffer'Read(Stream(f_in), b);
exception
  when Ada.Streams.
    Stream_IO.End_Error => null;
-- Nothing bad, just some garbage in
-- the buffer after end of compressed
-- code
end Read;
procedure Write( b: in Buffer ) is
begin
  Buffer'Write(Stream(f_out), b);
end Write;

```

Bad luck, it is as slow as doing I/O's with single bytes and Sequential_IO! But if it is slow by receiving/sending a whole buffer, how to make it faster? Now someone (in a slightly different context) came with this (call it variant 2):

```

procedure Read( b: out Buffer ) is
use Ada.Streams;
  First : constant
    Stream_Element_Offset :=
      Stream_Element_Offset(b'First);
  Last : Stream_Element_Offset:=
    Stream_Element_Offset(b'Last);
  SE_Buffer : Stream_Element_Array(
    First..Last);
begin
  Read(Stream(f_in).all,
    SE_Buffer, Last);
for i in First..Last loop
  b(Natural(i)):=
    Unsigned_8(SE_Buffer(i));
end loop;
end Read;
procedure Write( b: in Buffer ) is
use Ada.Streams;
  First : constant
    Stream_Element_Offset:=
      Stream_Element_Offset(b'First);
  Last : constant
    Stream_Element_Offset:=
      Stream_Element_Offset(b'Last);
  SE_Buffer : Stream_Element_Array(
    First..Last);
begin
for i in SE_Buffer'Range loop
  SE_Buffer(i):=
    Stream_Element(b(Natural(i)));
end loop;
  Write(Stream(f_out).all, SE_Buffer);
end Write;

```

Naively, you would say it is even slower: you do even more by copying a buffer into another one, right?

Indeed, not at all, it is *lots* faster (on GNAT and ObjectAda)!

To give an idea, the variant 1 applied to a bzip2 decompressor makes it 4x slower than the C version, and variant 2 makes it only 7% slower! With only I/O (like copying a file) you would get an even much larger difference.

Now, it raises some questions:

Is there maybe a reason in the RM why the 'Read and 'Write have to be that slow?

Or are these two compilers lazy when compiling these attributes ?

Should I bug AdaCore about that, then?

Do some other compilers do it better?

*From: Georg Bauhaus
Date: Fri, 30 Oct 2009 04:36:27 +0100
Subject: Re: Performance of the Streams*

*'Read and 'Write
Newsgroups: comp.lang.ada*

[...]

I finally thought that the above procedures are faster than 'Read or 'Write because the latter are defined in terms of stream elements:

When there is a composite object like b : Buffer and you 'Write it, then for each component of b the corresponding 'Write is called. This then writes stream elements, probably calling Stream_IO.Write or some such in the end. So Write from above appears closer to writing bulk loads of stream elements than a bulk load of 'Writes can be.

Copying buffers does not matter in comparison to the needs of I/O (on PCs).

*From: Gautier de Montmollin
<gdemont@users.sourceforge.net>
Date: Fri, 30 Oct 2009 02:13:19 -0700 PDT
Subject: Re: Performance of the Streams
'Read and 'Write
Newsgroups: comp.lang.ada*

[...]

Sure, it is the safe way: write records field by field, arrays element by element (that recursively). The compiler avoids problems with non-packed data. Nothing against that. The general case is well done, fine. But the compiler could have a look at the type left to the attribute and in such a case (an array of Unsigned_8, or a String) say: "Gee! that type Buffer is coincidentally the same as Stream_Element_Array, then I take the shortcut to generate the code to write the whole buffer and, this time, not the code to write it element by element".

> Copying buffers does not matter in comparison to the needs of I/O (on PCs).

Right. Variant 2 works fine, but it is an heavy workaround in terms of source code. Especially for mixed type I/O with plenty of String'Write and others, you would not want to put the kind of Variant 2 code all over the place. It would be a lot better that compilers are able to take selectively the shortcut form for the attributes.

*From: Randy Brukardt
<randy@rrsoftware.com>
Date: Mon, 2 Nov 2009 15:37:48 -0600
Subject: Re: Performance of the Streams
'Read and 'Write
Newsgroups: comp.lang.ada*

[...]

IMHO, Ada compilers should do that. (There's specifically a permission to do this optimization in Ada 2005: 13.13.2(56/2).) That's an integral part of the stream attribute implementation on Janus/Ada. (Disclaimer: the entire stream attribute implementation on Janus/Ada doesn't work right, quite probably because it is too complicated. So perhaps there is a

reason that other Ada compilers don't do that. :) Note, however, that it is pretty rare that you could actually do that (only about 15% of the composite types I've seen in Janus/Ada would qualify). So I'm not surprised that implementers have left that capability out in favor of things that happen more often.

*From: Gautier de Montmollin
<gdemont@users.sourceforge.net>
Date: Mon, 2 Nov 2009 14:16:30 -0800 PST
Subject: Re: Performance of the Streams
'Read and 'Write
Newsgroups: comp.lang.ada*

> [...] Note, however, that it is pretty rare that you could actually do that (only about 15% of the composite types I've seen in Janus/Ada would qualify).

Sure - but imagine that these 15% might transport 95% of the information. It could happen, couldn't it?

And if type T qualifies, a record type R with fields of types T,U,V (U and V not qualifying) will be also transmitted faster, an array of R will also go faster, and so on...

> So I'm not surprised that implementers have left that capability out in favor of things that happen more often.

I am not surprised either...

*From: Jeffrey R. Carter
<jrcarter@acm.org>
Date: Thu, 29 Oct 2009 17:39:39 -0700
Subject: Re: Performance of the Streams
'Read and 'Write
Newsgroups: comp.lang.ada*

[...]

And if you overlay the Stream_Element_Array onto the Buffer, thus eliminating the copying and the stream attribute operations?

*From: Jeffrey R. Carter
<spam.jrcarter.not@spam.acm.org>
Subject: Re: Performance of the Streams
'Read and 'Write
Date: Fri, 30 Oct 2009 12:12:18 -0700
Newsgroups: comp.lang.ada*

> With an Unchecked_conversion ?

No, that still does a copy.

Type Buffer, as a simple array of bytes, should have Buffer'Component_Size = Unsigned_8'Size by default; but you can specify it if you're paranoid. If you're really paranoid, you can add a test that Unsigned_8'Size = Stream_Element'Size, which you seem to be assuming. Then

procedure Put (B : in Buffer) is

-- Terrible naming scheme.

subtype Buffer_Stream is

Stream_Element_Array (
1 .. B'Length);

S : Buffer_Stream;

for S'Address **use** B'Address;

pragma Import (Ada, S);

begin -- Put

Write (S);

end Put;

*From: Randy Brukardt
<randy@rsoftware.com>
Date: Mon, 2 Nov 2009 15:32:51 -0600*

*Subject: Re: Performance of the Streams
'Read and 'Write*

Newsgroups: comp.lang.ada

> No, that still does a copy.

It doesn't have to, there is a permission to avoid copying in 13.9(12). So it depends on what the compiler is able to do optimization-wise.

*From: Gautier de Montmollin
<gdemont@users.sourceforge.net>
Date: Sun, 1 Nov 2009 13:38:44 -0800 PST
Newsgroups: comp.lang.ada*

*Subject: Re: Performance of the Streams
'Read and 'Write*

> If you're really paranoid, you can add a test that Unsigned_8'Size = Stream_Element'Size, which you seem to be assuming.

Yet a bit more paranoid: checking the size of arrays!

workaround_possible: Boolean;

procedure Check_workaround is

test_a: **constant** Byte_Buffer(1..10):=
(others => 0);

test_b: **constant** Ada.Streams.

Stream_Element_Array(1..10):=

(others=> 0);

begin

workaround_possible:=

test_a'Size = test_b'Size;

end Check_workaround;

It's the code I've put into Zip-Ada - big success!

Conference Calendar

Dirk Craeynest

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

This is a list of European and large, worldwide events that may be of interest to the Ada community. Further information on items marked ♦ is available in the Forthcoming Events section of the Journal. Items in larger font denote events with specific Ada focus. Items marked with ☺ denote events with close relation to Ada.

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

2010

- ☺ January 20-22 **37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'2010)**, Madrid, Spain. Topics include: all aspects of programming languages and systems, with emphasis on how principles underpin practice.
- ☺ January 19 **4th ACM SIGPLAN Workshop on Programming Languages meets Program Verification (PLPV'2010)**. Topics include: research at the intersection of programming languages and program verification; attempts to reduce the burden of program verification by taking advantage of particular semantic and/or structural properties of the programming language; all aspects, both theoretical and practical, of the integration of programming language and program verification technology.
- January 25-27 **5th International Conference on High Performance and Embedded Architectures and Compilers (HiPEAC'2010)**, Pisa, Italy. Topics include: Compilation techniques for embedded processors; Compilation and runtime support for multi- and many-core architectures; Tools and techniques for simulation and performance analysis; Tools for analysis, design, testing and implementation of embedded systems; etc.
- ☺ January 23 **HiPEAC2010 - 2nd Workshop on GCC Research Opportunities (GROW'2010)**. Topics include: current challenges in research and development of compiler analyses and optimizations based on the free GNU Compiler Collection (GCC). Deadline for early registration: January 6, 2010.
- ☺ Feb 03-04 **2nd International Symposium on Engineering Secure Software and Systems (ESSoS'2010)**, Pisa, Italy. Topics include: security architecture and design for software and systems, systematic support for security best practices, programming paradigms for security, processes for the development of secure software and systems, etc. Deadline for early registration: January 10, 2010.
- February 15-18 **4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'2010)**, Krakow, Poland. Includes track on: Software Engineering for Distributed Systems.
- ☺ Feb 15 **International Workshop on Multi-Core Computing Systems (MuCoCoS'2010)**. Topics include: multi-core embedded systems; programming languages and models; applications for multi-core systems; performance modeling and evaluation of multi-core systems; design space exploration; tool-support for multi-core systems; compilers, runtime and operating systems; etc.
- February 17-19 **18th Euromicro International Conference on Parallel, Distributed and network-based Processing (PDP'2010)**, Pisa, Italy. Topics include: Parallel Computer Systems (embedded parallel and distributed systems, fault-tolerance, multi/many core systems, ...); Models and Tools for Parallel Programming Environments: Advanced Applications (numerical applications with multi-level parallelism, real time distributed applications, distributed business applications, ...); Languages, Compilers and Runtime Support Systems (parallel languages, object-oriented languages, dependability issues, scheduling, ...); etc.

- March 09-11 16th French-speaking **Conference on Object-Oriented Languages and Models (LMO'2010)**, Pau, France.
- March 09-12 23rd IEEE-CS **Conference on Software Engineering Education and Training (CSEET'2010)**, Pittsburgh, PA, USA. Theme: "Bridging the Gap between Academia and Industry in Software Engineering Education and Training". Topics include: Curriculum and teaching materials, Software engineering professionalism, Internship and projects for students and graduates, Case studies of educational or training practices, Industry-academia collaboration models, etc.
- ☺ March 10-13 41st ACM **Technical Symposium on Computer Science Education (SIGCSE'2010)**, Milwaukee, Wisconsin, USA.
- March 15-18 14th European **Conference on Software Maintenance and Reengineering (CSMR'2010)**, Madrid, Spain. Topics include: Experience reports and empirical studies on maintenance, reengineering, and evolution; Description of education-related issues to evolution, maintenance and reengineering; Mechanisms and techniques for reengineering systems as services; etc.
- March 20-28 **European Joint Conferences on Theory and Practice of Software (ETAPS'2010)**, Paphos, Cyprus. Events include: FOSSACS, Foundations of Software Science and Computation Structures; FASE, Fundamental Approaches to Software Engineering; ESOP, European Symposium on Programming; CC, International Conference on Compiler Construction; TACAS, Tools and Algorithms for the Construction and Analysis of Systems.
- ☺ March 21 **Programming Language Approaches to Concurrency and communication-cEntric Software (PLACES'2010)**. Topics include: the general area of foundations of programming languages for concurrency, communication and distribution, such as language design and implementations for communications and/or concurrency, program analysis, multicore programming, concurrent data types, integration of sequential and concurrent programming, etc. Deadline for submissions: January 15, 2010 (abstracts).
- March 27 2nd **Workshop on Generative Technologies (WGT'2010)**. Topics include: Generative programming, metaprogramming; Analysis of language support for generative programming; Case Studies and Demonstration Cases; etc.
- March 27 7th International **Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA'2010)**. Topics include: Software quality attributes such as reliability, performance, or security; Interface compliance; Approaches for correctness by construction; Static and dynamic analysis; Runtime management of applications; etc.
- March 27-28 10th **Workshop on Language Descriptions, Tools and Applications (LDTA'2010)**. Topics include: applications of and tools for meta programming in a broad sense, such as Program analysis, transformation, generation and verification; Reverse engineering and reengineering; Refactoring and other source-to-source transformations; Language definition and language prototyping; Debugging, profiling and testing; etc.
- March 27-28 8th **Workshop on Quantitative Aspects of Programming Languages (QAPL'2010)**. Topics include: probabilistic, timing and general quantitative aspects in Language design, Multi-tasking systems, Language expressiveness, Verification, Time-critical systems, Safety, Embedded systems, Program analysis, Risk and hazard analysis, Scheduling theory, Distributed systems, Model-checking, Security, Concurrent systems, etc.
- March 22-26 25th ACM **Symposium on Applied Computing (SAC'2010)**, Sierre and Lausanne, Switzerland.
- ☺ Mar 22-26 **Track on Object-Oriented Programming Languages and Systems (OOPS'2010)**. Topics include: Language design and implementation; Type systems, static analysis, formal methods; Integration with other paradigms; Components and modularity; Distributed, concurrent or parallel systems; Interoperability, versioning and software adaptation; etc.
- ☺ Mar 22-26 **Track on Software Engineering (SE'2010)**. Topics include: technologies, theories, and tools used for producing highly dependable software more effectively and efficiently;

such as Safety and Security; Dependability and Reliability; Fault Tolerance and Availability; Architecture, Framework, and Design Patterns; Standards; Maintenance and Reverse Engineering; Verification, Validation, and Analysis; Formal Methods and Theories; Component-Based Development and Reuse; Empirical Studies, and Industrial Best Practices; etc.

☺ Mar 22-26 **Track on Real-Time Systems (RTS'2010)**. Topics include: all aspects of real-time systems design, analysis, implementation, evaluation, and case-studies, including scheduling and schedulability analysis; worst-case execution time analysis; modeling and formal methods; validation techniques; reliability; compiler support; component-based approaches; middleware and distribution technologies; programming languages and operating systems; embedded systems; etc.

☺ Mar 22-26 **Track on Programming Languages (PL'2010)**. Topics include: Compiling Techniques, Formal Semantics and Syntax, Garbage Collection, Language Design and Implementation, Languages for Modeling, Model-Driven Development and Model Transformation, New Programming Language Ideas and Concepts, Practical Experiences with Programming Languages, Program Analysis and Verification, Programming Languages from All Paradigms, etc.

Mar 22-26 **Track on Software Verification and Testing (SVT'2010)**. Topics include: development of technologies to improve the usability of formal methods in software engineering, tools and techniques for verification of large scale software systems, real world applications and case studies applying software verification, static and run-time analysis, correct by construction development, software certification and proof carrying code, etc.

March 22-26 17th **IEEE International Conference and Workshops on the Engineering of Computer Based Systems (ECBS'2010)**, Oxford, UK. Topics include: Component-Based System Design; Design Evolution; Distributed Systems Design; ECBS Infrastructure (Tools, Environments); Education & Training; Embedded Real-Time Software Systems; Integration Engineering; Model-Based System Development; Modelling and Analysis of Complex Systems; Open Systems; Reengineering & Reuse; Reliability, Safety, Dependability, Security; Standards; Verification & Validation; etc. Deadline for early registration: February 22, 2010.

March 24-26 15th **IEEE International Conference on the Engineering of Complex Computer Systems (ICECCS'2010)**, Oxford, UK. Topics include: Verification and validation, Reverse engineering and refactoring, Design by contract, Safety-critical & fault-tolerant architectures, Real-time and embedded systems, Tools and tool integration, Industrial case studies, etc.

April 06-09 3rd **IEEE International Conference on Software Testing, Verification and Validation (ICST'2010)**, Paris, France. Topics include: Verification & validation, Quality assurance, Empirical studies, Inspections, Tools, Embedded software, Novel approaches to software reliability assessment, etc.

April 06-09 21st **Australian Software Engineering Conference (ASWEC'2010)**, Auckland, New Zealand. Topics include: Empirical Research in Software Engineering; Formal Methods; Legacy Systems and Software Maintenance; Measurement, Metrics, Experimentation; Object and Component-Based Software Engineering; Open Source Software Development; Quality Assurance; Real-Time and Embedded Software; Software Design and Patterns; Software Engineering Education; Software Re-use and Product Development; Software Risk Management; Software Security, Safety and Reliability; Software Verification and Validation; Software Vulnerabilities; Standards and Legal Issues; Testing, Analysis and Verification; etc.

April 13-15 2nd **NASA Formal Methods Symposium (NFM'2010)**, Washington, D.C., USA. Topics include: Formal verification, including theorem proving, model checking, and static analysis; Model-based development; Techniques and algorithms for scaling formal methods, such as parallel and distributed techniques; Empirical evaluations of formal methods techniques for safety-critical systems; etc. Deadline for submissions: January 8, 2010 (abstracts), January 15, 2010 (papers).

☺ April 13-16 5th **European Conference on Computer Systems (EuroSys'2010)**, Paris, France. Topics include: various issues of systems software research and development, such as systems aspects of Dependable

computing, Distributed computing, Parallel and concurrent computing, Programming-language support, Real-time and embedded computing, Security, etc.

- April 13-16 ACM-BCS **Visions of Computer Science conference** (Visions'2010), Edinburgh, UK. Topics include: Programming Methods and Languages; Software Engineering, and System Design Tools; Distributed and Pervasive Systems; Robotics; Medical Applications; etc.
- April 15-16 2nd **International Workshop on Software Engineering for Resilient Systems** (SERENE'2010), London, UK. Topics include: methods and tools that ensure resilience to faults, errors and malicious attacks; Requirements, software engineering & re-engineering for resilience; Verification and validation of resilient systems; Error, fault and exception handling in the software life-cycle; Frameworks, patterns and software architectures for resilience; etc.
- ☺ April 19-23 24th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2010), Atlanta, Georgia, USA. Topics include: Parallel and distributed algorithms; Applications of parallel and distributed computing; Parallel and distributed software, including parallel and multicore programming languages and compilers, runtime systems, middleware, libraries, parallel programming paradigms, programming environments and tools, etc.
- ☺ April 19 15th **International Workshop on High-Level Parallel Programming Models and Supportive Environments** (HIPS'2010). Topics include: all areas of parallel applications, language design, compilers, run-time systems, and programming tools; such as New programming languages and constructs for exploiting parallelism and locality; Experience with and improvements for existing parallel languages and run-time environments; Parallel compilers, programming tools, and environments; Programming environments for heterogeneous multicore systems; etc.
- April 26-29 22nd Annual **Systems and Software Technology Conference** (SSTC'2010), Salt Lake City, Utah, USA.
- ☺ April 27 EDCC2010 - **Workshop on Critical Automotive applications: Robustness and Safety** (CARS'2010), Valencia, Spain. Topics include: design, implementation and operation of critical automotive applications and systems, with particular emphasis on dependability issues, software engineering for robustness, security and safety issues, real-time embedded systems technologies, architectural solutions and development processes for dependable automotive embedded systems. Deadline for submissions: January 20, 2010.
- ☺ May 02-08 32nd **International Conference on Software Engineering** (ICSE'2010), Cape Town, South Africa. Topics include: Engineering of distributed/parallel software systems; Engineering of embedded and real-time software; Engineering secure software; Patterns and frameworks; Programming languages; Reverse engineering and maintenance; Software architecture and design; Software components and reuse; Software dependability, safety and reliability; Software economics and metrics; Software tools and development environments; Theory and formal methods; etc.
- May 31 – June 02 10th **International Conference on Computational Science** (ICCS'2010), Amsterdam, The Netherlands. Topics include: recent developments in methods and modelling of complex systems for diverse areas of science, advanced software tools, etc. Deadline for submissions: January 1, 2010 (full papers). Deadline for early registration: March 31, 2010.
- May 31 3rd **International Workshop on Software Engineering for Computational Science and Engineering** (SECSE'2010). Topics include: Lessons learned from the development of CSE applications; The use of empirical studies to better understand the environment, tools, languages, and processes used in CSE application development and how they might be improved; etc. Deadline for submissions: January 19, 2010 (papers).
- May 31 7th **International Workshop on Practical Aspects of High-level Parallel Programming** (PAPP'2010). Topics include: high-level parallel language design, implementation and optimisation; modular, object-oriented, functional, logic, constraint programming for parallel, distributed and grid computing systems; industrial uses of a high-level parallel language; etc. Deadline for submissions: January 11, 2010 (full papers).
- May 31 – June 02 10th Annual **International Conference on New Technologies of Distributed Systems** (NOTERE'2010), Tozeur, Tunisia. Topics include: Domain Specific languages for distributed systems;

Reliability and scalability of distributed systems; Modeling, Formal and Semi-formal methods, and tools for distributed systems; Software and middleware for embedded distributed systems and their applications; etc. Deadline for submissions: January 20, 2010 (research papers).

- ☺ June 01-04 **DAta Systems In Aerospace (DASIA'2010)**, Budapest, Hungary.
- June 14-15 2nd **USENIX Workshop on Hot Topics in Parallelism (HotPar'2010)**, Berkeley, CA, USA. Topics include: the broad impact of multicore computing in all fields, including application design, languages and compilers, systems, and architecture. Deadline for position paper submissions: January 24, 2010.
- ♦ June 14-18 **15th International Conference on Reliable Software Technologies - Ada-Europe'2010**, Valencia, Spain. Sponsored by Ada-Europe, in cooperation with ACM SIGAda. Deadline for submissions: January 11, 2010 (industrial presentations).
- June 16-18 **Code Generation 2010**, Cambridge, UK. Topics include: Model-driven software development, Tool and technology development and adoption, Code Generation and Model Transformation tools and approaches, Defining and implementing modelling languages, Language evolution and modularization, Case studies, etc. Deadline for submissions: January 15, 2010.
- ☺ June 21-23 **Automotive - Safety & Security 2010**, Stuttgart, Germany. Organized by Gesellschaft für Informatik mit den Fachgruppen Ada, etc, and Ada-Deutschland. Topics include (in German): Zuverlässigkeit und Sicherheit für fahrbetriebs-kritische Software und IT-Systeme; Evaluation und Zertifizierung von Sicherheitseigenschaften automobiler Firmware/Software; Multi-Core-Architekturen; Zuverlässige Echtzeit-Betriebssysteme; Fortschritte bei Normen und Standardisierungen; etc. Deadline for submissions: January 14, 2010 (full papers).
- June 21-23 **AMAST2010 - 10th International Conference on Mathematics of Program Construction (MPC'2010)**, Québec City, Canada. Topics of interest range from algorithmics to support for program construction in programming languages and systems, such as type systems, program analysis and transformation, programming-language semantics, security, etc.
- ☺ June 21-25 **24th European Conference on Object Oriented Programming (ECOOP'2010)**, Maribor, Slovenia. Topics include: research results or experience in all areas relevant to object technology, including work that takes inspiration from, or builds connections to, areas not commonly considered object-oriented; such as: Analysis, design methods and design patterns; Concurrent, real-time or parallel systems; Distributed systems; Language design and implementation; Programming environments and tools; Type systems, formal methods; Compatibility, software evolution; Components, modularity; etc.
- June 21-25 **10th International Conference on Application of Concurrency to System Design (ACSD'2010)**, Braga, Portugal. Topics include: (Industrial) case studies of general interest, gaming applications, consumer electronics and multimedia, automotive systems, (bio-)medical applications, internet and grid computing, ...; Synthesis and control of concurrent systems, (compositional) modelling and design, (modular) synthesis and analysis, distributed simulation and implementation, ...; etc. Deadline for submissions: January 10, 2010 (papers).
- June 26-30 **15th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'2010)**, Ankara, Turkey.
- ☺ Jun 28 – Jul 02 **48th International Conference Objects, Models, Components, Patterns (TOOLS Europe'2010)**, Málaga, Spain. Topics include: Object technology, including programming techniques, languages, tools; Distributed and concurrent object systems; Real-time object-oriented programming and design; Experience reports, including efforts at standardisation; Applications to safety- and security-related software; Trusted and reliable components; Domain specific languages and language design; Language implementation techniques, compilers, run-time systems; Practical applications of program verification and analysis; etc. Deadline for submissions: January 22, 2010 (papers).
- July 05-12 **37th International Colloquium on Automata, Languages and Programming (ICALP'2010)**, Bordeaux, France.
- July 12-14 **2010 International Conference on Software Engineering Theory and Practice (SETP'2010)**, Orlando, Florida, USA. Topics include: Software development, maintenance, and other areas of software engineering and related topics.

- July 15-19 22nd **International Conference on Computer Aided Verification (CAV'2010)**, Edinburgh, UK. Topics include: Algorithms and tools for verifying models and implementations, Program analysis and software verification, Applications and case studies, Verification in industrial practice, etc. Deadline for submissions: January 11, 2010 (abstracts), January 15, 2010 (papers).
- July 22-24 5th **International Conference on Software and Data Technologies (ICSOFT'2010)**, Athens, Greece. Topics include: Software Engineering, Programming Languages, Distributed and Parallel Systems, etc. Deadline for submissions: February 1, 2010 (regular papers).
- July 25-28 29th Annual ACM SIGACT-SIGOPS **Symposium on Principles of Distributed Computing (PODC'2010)**, Zurich, Switzerland. Topics include: multiprocessor and multi-core architectures and algorithms; synchronization protocols, concurrent programming; fault-tolerance, reliability, availability; middleware platforms; distributed data management; security in distributed computing; specification, semantics, verification, and testing of distributed systems; etc. Deadline for submissions: February 10, 2010 (abstracts), February 17, 2010 (papers), April 27, 2010 (brief announcements).
- ☺ Aug 31 – Sep 03 16th **International European Conference on Parallel and Distributed Computing (Euro-Par'2010)**, Ischia, Italy. Topics include: all aspects of parallel and distributed computing, such as Support tools and environments, Scheduling, High performance compilers, Distributed systems and algorithms, Parallel and distributed programming, Multicore and manycore programming, Theory and algorithms for parallel computation, etc. Deadline for submissions: January 31, 2010 (abstracts), February 7, 2010 (full papers), March 1, 2010 (workshops).
- ☺ Sep 13-16 39th **International Conference on Parallel Processing (ICPP'2010)**, San Diego, California, USA. Topics include: compilers and languages, etc. Deadline for submissions: February 24, 2010.
- September 20-22 15th European **Symposium on Research in Computer Security (ESORICS'2010)**, Vouliagmeni, Athens, Greece. Topics include: Accountability, Information Flow Control, Formal Security Methods, Language-based Security, Security Verification, etc. Deadline for submissions: April 1, 2010.
- ♦ Oct 24-28 ACM SIGAda Annual **International Conference on Ada and Related Technologies (SIGAda'2010)**, Fairfax, Virginia (Washington DC Area), USA. Sponsored by ACM SIGAda, in cooperation with SIGBED, SIGCAS, SIGCSE, SIGPLAN, Ada-Europe, and the Ada Resource Association. Deadline for submissions: June 30, 2010 (technical articles, extended abstracts, experience reports, panel sessions, industrial presentations, workshops, tutorials).
- November 08-12 13th **Brazilian Symposium on Formal Methods (SBMF'2010)**, Natal, Rio Grande do Norte, Brazil. Topics include: Formal aspects of popular languages and methodologies; Logics and semantics of programming and specification languages; Type systems in computer science; Formal methods integration; Code generation; Formal design methods; Abstraction, modularization and refinement techniques; Techniques for correctness by construction; Formal methods and models for real-time, hybrid and critical systems; Models of concurrency, security and mobility; Theorem proving; Static analysis; Software certification; Teaching of, for and with formal methods; Experience reports on the use of formal methods; Industrial case studies; Tools supporting the formal development of computational systems; Development methodologies with formal foundations; etc. Deadline for submissions: June 10, 2010 (papers).
- December 10 Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!



**15th International Conference on
RELIABLE SOFTWARE TECHNOLOGIES
ADA-EUROPE 2010**

VALENCIA, SPAIN, 14-18 JUNE



In cooperation with
ACM SIGAda



UNIVERSIDAD
POLITECNICA
DE VALENCIA

<http://www.ada-europe.org/conference2010>

ADVANCE INFORMATION

The 15th International Conference on Reliable Software Technologies – Ada-Europe 2010 will take place in Valencia, Spain, on 14-18 June 2010. The conference has established itself as an international forum for providers, practitioners and researchers into reliable software technologies. Following tradition, the conference will span a full week, with a three-day technical program at its centre from Tuesday to Thursday accompanied by vendor exhibitions, and a string of parallel tutorials on Monday and Friday.

ABOUT THE VENUE

Valencia, situated on the Mediterranean coast of eastern Spain, is the capital city of the autonomous region Comunidad Valenciana. It has a population of around 800,000 inhabitants. Not many cities are capable of so harmoniously combining a fine array of sights from the distant past with innovative constructions now being erected. Valencia, founded in 138 BC, is one of these fortunate few. From the remains of the Roman forum located in today's Plaza de la Virgen to the emblematic City of Arts and Sciences, this town has transformed its physiognomy over the years while preserving its monuments from the past.

Sightseeing around the city begins in the old quarter, where the conference venue is located. Still standing as proof of the old defending wall are the graceful Torres de Serranos, the spacious Torres de Quart and some remains of the apron wall in the basement of the Valencia institute of Modern Arts. The Gothic building of La Lonja was declared heritage of humanity by UNESCO. It features a beautiful columned room where the old tables on which trading transactions were finalized are still in use today. On the old riverbed of the river Turia lie the nursery gardens, along with the Fine Arts Museum and the modern part of the city. Life in the city spreads down to the seafront with the harbor and the beaches of Las Arenas and La Malvarrosa.



INVITED SPEAKERS

Three eminent keynote speakers have been selected to open each day of the core conference program:

- **Pedro Albertos** (Universidad Politécnica de Valencia, Spain), a worldwide expert in Automatic Control, will explore the relationship between implementation and performance of control algorithms in a talk entitled *Control Co-Design: Algorithms and their Implementation*.
- **Theodore Baker** (Florida State University, USA), a leading researcher in Ada and Real-Time systems, will examine the state of the art in multiprocessor real-time scheduling in his talk *What to Make of Multicore Processors*.
- **James Sutton** (Lockheed Martin, USA), a worldwide expert in programming technologies, will explore how Ada is prepared for the so-called 3.0 World, a world that makes peace with complexity and chaos, and learns to use them to its advantage. That will be in his talk entitled *Ada: Made for the 3.0 World*.

TUTORIALS

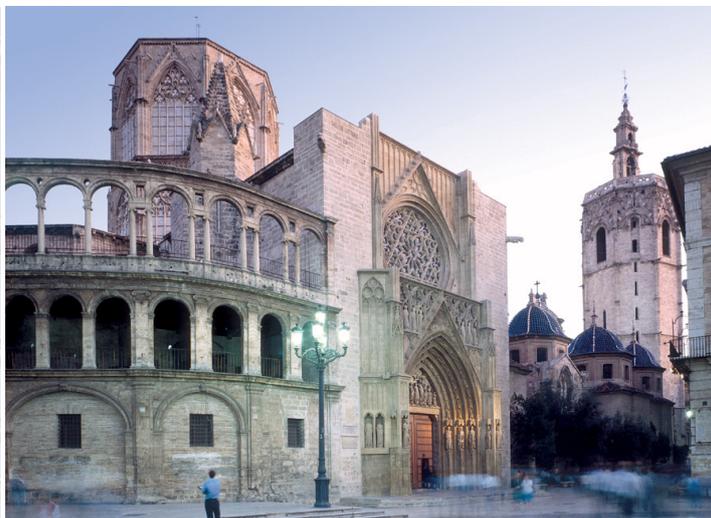
Attendees will have a varied choice of half- and full-day tutorials that will be offered on Monday and Friday before and after the central days of the conference. Tutorials consist of monographic courses, given by recognized experts in their respective fields, which deal with up to date technologies for the development of reliable software.

SOCIAL PROGRAM

The social program will schedule two events: a welcome reception on Tuesday in the city center and a banquet dinner on Wednesday in a typical Masía —an old country house— at a very short distance from the city.

FURTHER INFORMATION

The conference web site at <http://www.ada-europe.org/conference2010.html> will be giving full and up to date details of the program and the venue, including travel advice, maps and a list of hotels close by. For Exhibiting and Sponsoring details please contact the Exhibition Chair, Ahlan Marriott, by email at Ada@white-elephant.ch. A sliding scale of sponsorship provides a range of benefits. All levels include display of your logo on the conference web site and in the program. The lowest level of support is very affordable.



Call for Technical Contributions – SIGAda 2010



**ACM Annual International Conference
on Ada and Related Technologies:
Engineering Safe, Secure, and Reliable Software**

Fairfax, Virginia (Washington DC Area), USA
October 24-28, 2010



Submission Deadline: June 25, 2010

Sponsored by ACM SIGAda

<http://www.acm.org/sigada/conf/sigada2010>

SUMMARY: Reliability, safety, and security are among the most critical requirements of contemporary software. The application of software engineering methods, tools, and languages all interrelate to affect how and whether these requirements are met.

Such software is in operation in many application domains. Much has been accomplished in recent years, but much remains to be done. Our tools, methods, and languages must be continually refined; our management process must remain focused on the importance of reliability, safety, and security; our educational institutions must fully integrate these concerns into their curricula.

The conference will gather industrial and government experts, educators, software engineers, and researchers interested in developing, analyzing, and certifying reliable, safe, long-lived, secure software. We are soliciting technical papers and experience reports with a focus on, or comparison with, Ada.

We are especially interested in experience in integrating these concepts into the instructional process at all levels.

POSSIBLE TOPICS INCLUDE BUT ARE NOT LIMITED TO:

- | | |
|---|--|
| <ul style="list-style-type: none"> • Challenges for developing reliable, safe, long-lived, secure software • Transitioning to Ada 2005 • Ada and SPARK in the classroom and student laboratory • Language selection for highly reliable systems • Mixed-language development • Use of high reliability subsets or profiles such as MISRA C, Ravenscar, SPARK • High-reliability standards and their conformance to DO-178B and preparing for DO-178C • Software process and quality metrics • System of Systems • Real-time networking/quality of service guarantees • Real-Time Parallel Processing | <ul style="list-style-type: none"> • Analysis, testing, and validation • Use of ASIS for new Ada tool development • High-reliability development experience reports • Static and dynamic analysis of code • Integrating COTS software components • System Architecture & Design • Information Assurance • Ada products certified against Common Criteria / Common Evaluation Methodology • Distributed systems • Fault tolerance and recovery • Performance analysis • Implementing Service Oriented Architecture • Embedded Hard Real-Time Systems |
|---|--|

KINDS OF TECHNICAL CONTRIBUTIONS:

TECHNICAL ARTICLES present significant results in research, practice, or education. Articles are typically 10-20 pages in length. These papers will be double-blind refereed and published in the Conference Proceedings and in ACM Ada Letters. The Proceedings will be entered into the widely-consulted ACM Digital Library accessible online to university campuses, ACM's 80,000 members, and the software community.

EXTENDED ABSTRACTS discuss current work for which early submission of a full paper may be premature. If your abstract is accepted, you will be expected to produce a full paper, which will appear in the proceedings. Extended abstracts will be double-blind refereed. In 5 pages or less, clearly state the work's contribution, its relationship with previous work by you and others (with bibliographic references), results to date, and future directions.

EXPERIENCE REPORTS present timely results on the application of Ada and related technologies. Submit a 1-2 page description of the project and the key points of interest of project experiences. Descriptions will be published in the final program or proceedings, but a paper will not be required.

PANEL SESSIONS gather a group of experts on a particular topic who present their views and then exchange views with each other and the audience. Panel proposals should be 1-2 pages in length, identifying the topic, coordinator, and potential panelists.

INDUSTRIAL PRESENTATIONS Authors of industrial presentations are invited to submit a short overview (at least 1 page in size) of the proposed presentation to the Industrial Committee Chair by August 1st 2010. The authors of selected presentations shall prepare a final short abstract and submit it to the Committee Chair by October 1st^h, 2010, aiming at a 20-minute talk. The authors of accepted presentations will be invited to submit corresponding articles for publication in the ACM Ada Letters.

WORKSHOPS are focused work sessions, which provide a forum for knowledgeable professionals to explore issues, exchange views, and perhaps produce a report on a particular subject. A list of planned workshops and requirements for participation will be published in the Advance Program. Workshop proposals, up to 5 pages in length, will be selected by the Program Committee based on their applicability to the conference and potential for attracting participants.

TUTORIALS offer the flexibility to address a broad spectrum of topics relevant to Ada, and those enabling technologies which make the engineering of Ada applications more effective. Submissions will be evaluated based on relevance, suitability for presentation in tutorial format, and presenter's expertise. Tutorial proposals should include the expected level of experience of participants, an abstract or outline, the qualifications of the instructor(s), and the length of the tutorial (half-day or full-day). Tutorial presenters receive complimentary registration to the other tutorials and the conference.

HOW TO SUBMIT: Send contributions by **June 25, 2010**, in Word, PDF, or text format as follows:

Technical Articles, Extended Abstracts, Experience Reports, and Panel Session Proposals to: Program Chair, Lt. Col. Jeff Boleng (Jeff.Boleng@usafa.edu)

Tutorial Proposals to: Tutorials Chair, Dr. Robert Pettit (RPettit@gmu.edu)

Industrial Presentations Proposals to: Industrial Committee Chair, Prof. Liz Adams (adamses@cs.jmu.edu)

FURTHER INFORMATION:

CONFERENCE GRANTS FOR EDUCATORS: The ACM SIGAda Conference Grants program is designed to help educators introduce, strengthen, and expand the use of Ada and related technologies in school, college, and university curricula. The Conference welcomes a grant application from anyone whose goals meet this description. The benefits include full conference registration with proceedings and registration costs for 2 days of conference tutorials/workshops. Partial travel funding is also available from AdaCore to faculty and students from GNAT Academic Program member institutions, which can be combined with conference grants. For more details visit the conference web site or contact Prof. Michael B. Feldman (mfeldman@gwu.edu)

OUTSTANDING STUDENT PAPER AWARD: An award will be given to the student author(s) of the paper selected by the program committee as the outstanding student contribution to the conference.

SPONSORS AND EXHIBITORS: Please contact Greg Gicca (Gicca@AdaCore.com) and Kristen Ferretti (kef@ocsystems.com) for information about becoming a sponsor and/or exhibitor at SIGAda 2010.

IMPORTANT INFORMATION FOR NON-US SUBMITTERS: International registrants should be particularly aware and careful about visa requirements, and should plan travel well in advance. Visit the conference website for detailed information pertaining to visas.

ANY QUESTIONS?:

Please submit your questions to Conference Chair Alok Srivastava (alok.srivastava@auatac.com) or Local Arrangements Co-Chairs Avtar Dhaliwal (avtar_dhaliwal@gencosystems.com) and Florence Gubanc (fgg@ocsystems.com).

Book review: Ada for Software Engineers, by Mordechai Ben-Ari

Tullio Vardanega

University of Padua, Department of Pure and Applied Mathematics, via Trieste 63, 35121 Padua, Italy; email: tullio.vardanega@math.unipd.it

1 Introduction

There is some irony in me writing a review for this book. Twofold irony, in fact. On the one hand, when I first eyed Mordechai Ben-Ari's book, fresh off the press, on one of the exhibitor's stalls at the Ada-Europe 2009 Conference in Brest, France, I said to myself: "I must read this book, for it's got all the right keywords in the title and I also trust the author". I did not grab the book then, for I wanted to play fair to the other conference attendees, and let them have their complimentary copy first. So I kept the resolve in the back of my mind and, as it often happens, daily routine made me stay very far from returning to it. On the other hand, I can now see that the favourable gaze I gave to the book was caused by me mistaking its purpose altogether. In fact I never thought myself much attracted to programming books, and thus I assumed Ben-Ari's book discussed how software engineering principles can be served by Ada, which was much better placed in the land of my favourites. Well, I was wrong in many ways, and I can now see it fully after reading Ben-Ari's book end to end. I must therefore offer my gratitude to the editor of the Ada User Journal – you don't know how relieved I am not to be talking to myself here anymore ☺ – for asking me to do this review. And here I am, writing the review for a book I much liked and thoroughly enjoyed, and that I badly wanted to read, just because I ignored or mistook its true subject.

Enough with reminiscence now, and on to the job.

You may have happened to hear or listen or possibly even take part in debates over what programming language should be taught in Universities and which should be taught first. For some reason, even before becoming an educator myself, I thought that University education should always put concepts, paradigms and methodologies in the foreground, and place technology-specific discourse in the background. I was lucky and grateful that my own University education was precisely in that line. Of course I understand that the industry dominant view of education leans more toward acquiring technical skills than on mastering the fundamentals, on the (short-sighted) assumption that the hard measure of productivity is the number of lines of code written per unit of time. The net result of skipping over the basics to jump onto specific high-level programming paradigms however is not especially encouraging and some fear – with good reason, I am afraid – that great damage is done to software engineering education by that course of action. However, as

for myself, I never wanted to join the fight for my favourite language to be taught first. Actually, I always argued, to employers and students alike, for the higher efficiency of starting from solid foundations before taking on a new programming language, than just developing language fluency on thin footing. The response I got then and often continue to get now is that industry cannot train their recruits to the programming language of choice, for this is the job of educators. Well, this is sad, isn't it. However, I know of industry who thinks exactly the converse and thus can console myself a little from the grief.

2 In the way of a review

The reader should not misunderstand me and think that I am going astray. No, I am not. But then, why I am discussing this? Because Ben-Ari's book matches my thinking in the regard of programming language education so very well, this of course be said without losing sense of proportion. If you asked me when I would place the teaching of Ada in a software engineering curriculum, I would then tell you that I would fit it no earlier than in the third or fourth year. I would consider basic knowledge of computer science and computer systems prerequisite, along with two to three years of programming experience. Only then, I think, learning Ada can be the aim instead of the means to a different end, like teaching programming principles. And guess what? I found these considerations almost verbatim under the "intended audience" paragraph at the start of Ben-Ari's book. So I said to myself, well, this must be my book and continued on reading light-heartedly.

I am now in my seventh year of lecturing in a computer science curriculum after my inception in the University profession. I have never taught a programming language class and never offered myself to teach one (you can guess what language it would be if I should). I only insisted that the programming language education of the curriculum had an escalating slant, so that more advanced paradigms, like object orientation, concurrency, distribution, systems programming and real-time, could be planted on a firmer basis. I am lucky that the current curriculum matches my wishes close enough. So I can occasionally refer to Ada in my software engineering class in the third year, with hope of making an impact, and then extensively use Ada in my class on concurrency and distribution in the fourth year, asking the students to learn the spans of Ada they need for my assignments, by contrasting it to the languages they were taught before.

Experience proves that this teaching strategy suits best the students who have grown fit for intellectual challenges – and some actually do – but causes headaches to those who view the world as the more popular languages depict it. Think of how students learn Java or C++, or even C#. They hardly ever read manuals, so probably then don't actually "learn", but rather memorize. Instead, or in fact, they go by examples and devour tutorials that help them construct working applications. Now, the world of Ada is so distant from this style that naïve students get frustrated. Ada's genotype is abhorrent of the hype and most of the Ada key figures are so busy doing important things (and this is literally true, really) that they don't have time to kill in developing basic tutorial-like materials for the uninitiated. So one is left to wonder where students can be referred to for filling in their Ada education gaps. Of course there are some valid sources around, but they look sparse to my eyes, or too Ada-centric. And being Ada-centric is not especially apt to win those who go to Ada by command – and feel going upstream there – and not by choice. So I would need something else, which I had not found yet.

Based on this experience, if I were to single out the characteristics that I deem vital for an effective Ada education, I would enumerate three:

- Explaining why things are the way they are in Ada, in terms of the language reference manual. This teaches students that programming languages should obey standard behaviour and that syntax serve to convey semantic information in manners that fit language design.
- Contrasting the Ada way of things to other, more popular languages that one may safely assume the

students already know. This helps students understand differences and value them for offering a different look to the same problems.

- Constructing case-study examples by using successive increments of language features. This helps students feel like having tutorial material, yet without the no-effort hype element of it.

I was amazed when I read Ben-Ari's own presentation of the structure of the book. What I read there and later found in the actual contents of the book – which proves that the author was true to his words – was the closest match to my requirements I could ever imagine.

I said I am not a fan of language textbooks – though I have always found John Barnes's books on Ada most preciously informative and beautifully written. However I think I have now found one that I can refer my students to with full empathy and trust that they can follow the trail that I would like them to tread when figuring how to think in Ada, hopefully before writing code.

At the time of writing, I have just finished my yearly class on concurrent and distributed systems, and given out the usual tough assignment. I know that students will feel compelled – no, not obliged, I hope – to try and use Ada to solve the problems I have posed them. I can predict that some will come back to me asking for guidance on how to do this and that in Ada and wondering why they can't do it the way Java does or C++. I now know that I will point them to Ben-Ari's book for a response. Perhaps in some time I may ask the Ada User Journal editor permission to report on how the book fared in the students' experience.

Experiences in Evaluating Ada with a Pilot Project

Maciej Sobczak

CERN, CH-1211 Geneve 23, Switzerland; email: Maciej.Sobczak@cern.ch

Abstract

This article describes author's experiences in evaluating and introducing Ada as a complementary programming language in a non-Ada environment in the context of large-scale distributed control system. The article presents the problem, motivations and a rather bitter outcome of the exercise.

1 Introduction

The European Organization for Nuclear Research (CERN) has managed to gain headlines during the last year thanks to its flagship project – LHC, the Large Hadron Collider. This project is described as the largest machine ever build on the planet and indeed many of its construction parameters are defining new limits in many engineering disciplines. From the perspective of the software community, the computing facilities at CERN are of particular interest and the control system that drives the whole machine is certainly interesting to those that deal with control, embedded, distributed and real-time systems.

The accelerator control system at CERN can be characterized by massive distribution with around 60_000 logical devices spread among almost 2_000 real-time servers (so-called front-ends) and accessed by tens of operator applications that are used for configuration of settings and monitoring of those devices. A logical device might represent a physical measurement, an actuator or a virtual data source that concentrates, filters or pre-computes the values from other sources – in all these cases logical devices are defined in a uniform and location-transparent name space with a single and unified method to follow from device name to actual implementation server. This makes the directory and name services crucial infrastructure components of the whole system.

Till the end of last year, the typical sequence of actions for any client application that wanted to access a given logical device was the following:

- Query the directory server for the device name – the directory server replied with the front-end server name that is exposing data and settings for the given device (many devices can be handled by a single server). The directory server uses the device dictionary from the main configuration database.
- Query the CORBA name server for the front-end server name – the name server replied with the IOR location of the server's main object. The name server used the IOR information that is automatically provided by each front-end server when it is started.

- Having the object reference, access the actual front-end server with the details of the requested operation and the logical device name.

In other words, the communication infrastructure relied heavily on the availability of two separate components (the device directory and the CORBA name server), both of which were implemented in Java and generated regular reliability and maintenance issues, both of them were also working as single instances – obviously a very weak point in the whole control system. In this context the decision was taken to provide a better directory and name lookup solution, which would guarantee the logical continuation of the service, but with higher reliability and potential performance improvements.

An important property of this project was that it was not bound to any existing technology or communication protocol and that it was under the responsibility of a single team. Such circumstances seemed to be a good opportunity to select the appropriate technology without any imposed constraints – and this is when the idea of using it as a pilot project for trying out and evaluating Ada was born.

2 Show that it is easy

The pilot project was meant to evaluate Ada as a prospective candidate for other future projects and as such it had to show that the new technology could be accommodated with as little investment as possible – with all possible meanings of “investment”, since every single cost of introducing it could be a potential argument against it. There were two major aspects that had to be addressed: the cost of acquiring and installing the necessary development tools and the cost of developers' training.

The choice of development tools was pretty obvious in the context of Linux as the standard software environment for all infrastructure-related installations – GNAT is a standard package in the Scientific Linux CERN, which is based on RedHat. The only potential problem was that the necessary certification and acceptance process for the whole operating system lags behind the recent developments and the only version of GNAT that was available on this system was 3.4.6, which is known to be less than perfect. Fortunately, it was not a problem in the context of this project.

The choice of GNAT and its inherent ability to link with the existing native libraries was also very important in the context of the database access. The SOCI-Ada library was used to handle that part of the project, and this was a natural extension of the existing use of the SOCI library in other applications that were implemented in C++. Thanks

to this there was no need to introduce any new database access method – obviously an important point in solving database access issues.

The potential cost of team training can be minimized with careful selection of the language elements that were used in the project – unrestricted use of advanced language features would make it more difficult for others to understand the code. Interestingly, the main recommendations of the Ravenscar profile, even though intended for specialized needs, seemed to provide a perfect way to simplify the language subset for this project. Static set of tasks that communicate only via protected objects was a proper model that was easy to comprehend without the necessity to learn the complete language. Similarly, the data structures used in the project were mainly statically constrained arrays and there was no need to complicate the code with dynamic memory management.

The choice of development tools, using the existing database access method and the discipline in keeping the code simple were important arguments to show that the use of new technology in this project was not a revolution and could be attempted with little cost.

3 Designing the new system

As already explained in the Introduction, the previous directory/name service was based on two separate components, one of them being a standard CORBA service. Such a setup forced the client applications to physically use separate network interactions for something that conceptually was a single logical service – resolving just a device name or just a front-end server name was not a major use-case and it was perceived as a lost opportunity to optimize the client-server interaction. Another optimization strategy that was unnecessarily difficult with this two-server setup was the concept of array queries, where a client application is interested in interaction with many logical devices (there are several cases with collective display applications that access up to one thousand logical devices at a time). Allowing such an application to completely resolve thousand device names in a single query instead of interacting with two separate services for each device name separately could dramatically improve the start-up time of such programs. Another important problem with the two-server setup was the inconsistent approach to data storage – the directory server used a relational database whereas the CORBA name server used a big local file in proprietary format – this inconsistency limited the maintenance and monitoring options.

A natural way to solve these problems was to merge the two services into a single one providing both device and server name resolutions in a single interaction.

The CORBA protocol was abandoned and a trivial text-based protocol was used instead. This move was criticized as unnecessary and a very important argument against it was related to the fact that all other components in the system would need to be modified in order to understand the new protocol, but on the other hand CORBA was previously responsible for another maintenance nightmare

related to the huge number of established connections that were kept alive without any purpose – a rather cumbersome property of the CORBA approach. Since the directory interactions are usually related to the initialization phase of any given component in the system, it was reasonable to close the connection on both sides just after the interaction. This resulted in the reduced number of active network connections from thousands to just a few that are currently active – a move that was very much appreciated by the network administrators.

Two important design decisions were directly related to the reliability of the new service:

- Introduction of the redundancy of installation – up to the database server. In other words, two instances of the directory server running on separate machines should be able to provide continuous service even with possible downtime of any single instance. The notion of redundancy was also introduced at the user side, where front-end servers were supposed to bind their locations at both directory instances (at least one completed interaction is required to consider the bind operation to be successful) and client applications randomly picked one of the two instances to execute their queries with potential failover to the other one. Even though the introduction of redundancy was motivated by the reliability concerns, it was actually useful mainly to allow comfortable rollout of improvements without interruption of the service. Obviously, an important aspect of the redundancy is the data synchronization – this was achieved with a single shared database. In other words, each directory instance writes new data to the database and periodically reads the whole set back, which ensures that sooner or later the information propagates from one instance to another. In any case, the propagation of information is of concern only when the new server location was bound to one of the instances while the other one was not active.
- Treating the database as a non-critical component – in other words, allowing the database to be unavailable for some period of time without affecting the ability of the directory server to provide its services. This decision was very important due to the fact that the central database server undergoes routine upgrades and patching several times per year and this should not have any impact on the critical infrastructure services. Since the database is used not only as a long-term storage but also as a data propagation channel between all directory instances, the server needs to be able to temporarily store the new information in the local file until the database is available again. It is worth to note that the local file store might potentially become a source of data inconsistency if one of the servers is temporarily unavailable at the same time when the database itself is unreachable, which would prevent the servers to exchange new data properly – in addition, in the same scenario the local store of the other server would become the only place where the new

information is stored. The probability and risks associated with this scenario were judged as acceptable and indeed the operational experience confirmed that further reliability improvements were not necessary.

The whole service is a natural place for the use of multitasking – a classical architecture for network servers was chosen with the following specialized tasks:

- Single acceptor task that waits for new incoming connections from clients and that hands them over to the worker tasks.
- Several worker tasks that wait for the availability of the “work item” and processing it in a loop. The number of tasks was set to 5 as a rough estimation of the needs – according to the log files, these 5 tasks were always able to handle the whole traffic in the control system.
- Database access task that asynchronously handles all database activity: reading the whole device dictionary, storing new server bindings and reading them back. The asynchronous nature of this task means that no other task waits for this one to execute any given work unit – instead, other tasks only notify that some type of access needs to be performed and the database access task at some point in time (usually immediately) does what was requested. The lazy and seemingly unpredictable nature of this task is what allows the whole server to survive the periods of database unavailability.
- Controller task (that is the environment task after executing all initialization work) that periodically triggers the database readout. It might be also a proper place to handle the collection and publication of various statistics.

Two major data structures that are shared between all these tasks are device and server maps, implemented as statically allocated arrays or appropriately defined records.

The following two tables illustrate conceptually both the device map and the server map.

Device map:

Device_Name	Server_Name
Termostat_1	Term_Srv_A
Termostat_2	Term_Srv_A
Power_Converter_123	Pow_Srv_P

Server map:

Server_Name	Server_Location
Term_Srv_A	IOR://432835534...
Pow_Srv_P	IOR://314245326...

In the most typical case, the client application is interested in obtaining the Server_Location (that is, the information

that allows to establish the physical connection with the target server) for the given Device_Name, which requires the availability of both of the above structures. The ability to perform such a query within a single interaction is the major added value with regard to the old solution.

It should be noted that the two data structures above are conceptually very simple, but allow for future growth in terms of new use-cases, which might include security protections against accidentally overwriting existing bindings, audit scans to check if the target servers are actually announcing themselves from the locations where they are supposed to be, statistical analysis of the target server reboot patterns and more. From the point of view of the client application, however, the combination of these two maps the most vital resource.

4 Feasibility Study and Demonstration

Since this pilot project was the first attempt to use Ada in the team, we needed to demonstrate that all technical aspects resulting from the design are well mastered and that no major surprises would threaten the project later on. Assuming the above design considerations, the following programming problems were identified:

- Reading and writing data structures from/to local files - Direct_IO was a proper choice for this.
- Writing formatted log files – Text_IO for file operation and Calendar for time reporting proved to be useful, with the small exception of the time formatting that was not available in Ada 95. For this reason the convenient Image routine that in Ada 2005 is available in Calendar.Formatting had to be implemented from scratch.
- Accessing the Oracle database – as already explained, the SOCI-Ada library was used to handle that part.
- Network operations – provided by GNAT.Sockets.

Several simple programs have been implemented to demonstrate that these programming problems are well mastered and that the whole project can be attempted without major risks.

5 Coding – get it done

The coding phase of this pilot project did not reveal any surprises thanks to the feasibility study and demonstration that was performed previously.

Just for the record, the implementation phase resulted in 16 Ada packages with total line count of ~5_000.

6 Demonstrate responsibility sharing

An important aspect of the whole exercise was to demonstrate that the new technology does not jeopardize our ability to maintain the software in the long term even with possible rotation of the team members. Without this demonstration it was clear that the idea of introducing new programming language would not be accepted based on valid management concerns. Therefore, part of the project was to involve somebody else and ask him to introduce

some modifications to the code as a mean to check if the responsibility for the whole can be passed or shared.

This part was surprisingly easy, contrary to the popular belief that Ada cannot be introduced to non-Ada programmers. Granted, the person who was involved had some limited exposure to the language as part of his university curriculum, but our perception was that it was not critical to guarantee his ability to understand and modify the existing code. The ease of responsibility sharing can be attributed to the fact that a reasonably small subset of the language was used in the implementation and all the programming concepts were immediately visible in the code.

Presumably it would not be that easy if some advanced language features were freely used in the code, but that possibility is inconclusive in the context of this exercise – after all, obfuscation of code can be achieved in any language.

7 Deployment

The pilot project was meant to evaluate Ada as a potential alternative technology for our developments. The only justification for such evaluation is the intention to improve the state of the art and to make a progress with relation to previous results. Needless to say, the motivation to try out Ada was to reduce the defect rate in the software that we produce and that on average is not very satisfactory.

How did it work in this pilot project? One thing can be firmly stated: the defect rate achieved in this project was much below the average and the major visible effect was that the complete server was put into production without any hassle and with immediate ability to handle the traffic of the whole distributed control system. There was no “infancy” stage and the system was mature from its first version. The new directory server is in operation from December last year and is successfully providing continuous service without any hiccups. This was an achievement that was very visible and acknowledged.

To be very honest, though, the zero-bug goal was not fully achieved and we had to face two embarrassing coding mistakes:

- The logging component automatically switched to another log file after reaching the given limit of entries in the current file – this way a series of log files is created instead of a huge and single one. The process of switching to another file involves closing the current file object (the type of this object is `Ada.Text_IO.File_Type`) and opening the same object for another physical file name – obviously a non-atomic sequence of actions that leaves a tiny window of opportunity for other tasks to see it in a state that does not allow proper use. The protection against this possibility was not implemented in the initial version, which resulted in a failure.
- The network component reported exceptions whenever the interaction with clients did not follow the expected path. These exceptions were handled in the outermost

exception handler, where the informative message was formatted and logged. The trap that we fell into is that in order to put as much useful information in the log message as possible, the `GNAT.Sockets.Get_Peer_Name` function was used to get the address of offending client. Surprisingly, this function itself resulted in exception if the client socket happened to be already closed at the time it was called – this new exception was not handled (it all happened already in the outermost handler, so there was no other enclosing handler), which silently killed the worker task.

Neither of these issues caused the interruption of the service thanks to the redundancy of the whole installation, but was anyway embarrassing – mainly because such bugs are not related to any programming language and therefore cannot be attributed to our problems in using the new language. On one hand - these were our own mistakes. On the other hand – Ada did not prevent us from making them. In retrospect, both faults were related to the fact that the respective language features (multitasking and exceptions) lead to execution paths and module interactions that are not explicit in code and therefore cannot be statically analyzed. This interesting observation proves that special language profiles and static contracts are justified when the ultimate reliability is expected.

To summarize – even with these two embarrassing issues the overall perception is that the new directory server is one of the most reliable components in the system and its implementation and deployment were surprisingly fluent. Personally, I consider it to be a very good indication of the ability of Ada to handle mission-critical infrastructure tasks. From the technical standpoint, the evaluation was positive.

This result was meant to be a strong argument for expanding our set of technologies and to introduce Ada as a first-class citizen in our control system. But...

8 Bitter outcome of the exercise

After officially presenting the project to the wider audience, which included senior engineers and other managers, it appeared that the idea of introducing the new programming language is not very well perceived. The major argument against it was the prevalent belief that Java is good enough as a technology for implementing all middle-tier components including the infrastructure services and that there is no need to extend our existing toolset. Another important argument was that in the working environment that involves tens of people there is a lot of value in standardization and consolidation of that toolset – a context where any new technology is obviously perceived as an “intruder” that can only encourage others to divert from the “standard” by introducing whatever fancy technology they like and therefore undermine the efforts to manage consistent technical culture in the whole development group.

The sad part of this outcome is not only that the pilot project did not achieve the positive perception of Ada,

which was the original intention of the author, but rather that after the discussion on introducing the new programming language was finally “settled”, the acceptance of Ada as a valid technology for the implementation of mission-critical components of the accelerator control system at CERN is not likely to happen anytime soon. An important part of this settlement is that even though nobody ever questioned the value and quality of the new directory server that is already in operation, it is supposed to be replaced with an implementation that conforms to the standard toolset.

The bitter lesson that was learnt is that the major obstacle in the wider adoption of Ada is not related to its technical

attributes, but rather to its ability to compete in the decision-making process that involves too many non-technical factors.

About the author

The author is working as a Middleware Team Leader for the accelerator control system at CERN, where he is responsible for the proper operation of the communication infrastructure. Apart from this work, he is a contributor of open-source software and provides independent consulting services.

You can contact him at <http://www.msobczak.com/> and at <http://www.inspirel.com/>

Couverture: an Innovative Open Framework for Coverage Analysis of Safety Critical Applications

Matteo Bordin, Cyrille Comar, Tristan Gingold, Jérôme Guitton, Olivier Hainque, Thomas Quinot

AdaCore, 46 rue d'Amsterdam, 75009 Paris (France);

{bordin, comar, gingold, gutton, hainque, quinot}@adacore.com

Julien Delange, Jérôme Hugues, Laurent Pautet

Télécom ParisTech, 46 rue Barrault, 75634 Paris Cedex 13 (France);

{julien.delange, laurent.pautet, jerome.hugues}@telecom-paristech.fr

Abstract

One key step in the development of safety-critical applications is the assessment of the quality of the verification strategy. In practice, structural coverage is the methodology used to ascertain the testing campaign well satisfy a given quality criteria. In this paper, we describe the possible strategies to measure structural coverage in a DO-178B context. After evaluating state-of-the-art approaches to measure structural coverage, we introduce Couverture, an innovative framework for coverage analysis exploiting a virtualized execution environment.

Keywords: structural coverage, DO-178B, MC/DC virtualization, Ada.

1 Introduction

The development of a high-integrity application usually requires a close interaction between the design and testing phases. System requirements are decomposed into high-level architecture, then into atomic modules; the latter are then finally implemented, possibly using a third generation programming language such as Ada or C/C++. A set of corresponding verification activities mirrors each decomposition step in the design phase: for example, acceptance testing corresponds to system design, integration testing to architectural design and unit testing to module design. The *test cases* are derived from the requirements themselves: the idea is to have the *specifications*, rather than the *implementation*, drive the testing strategy. This is the *V process model* applied to software engineering (see fig. 1).

One key step in the V development process is the assessment of the quality of the testing strategy. For software systems, *structural coverage* [8] is the canonical approach in practice. Structural coverage is the analysis of how an application is exercised by a testing campaign. The basic idea behind coverage analysis is to gain confidence in the testing process by checking that the testing suite exercises all *meaningful* constructs present in the application in a *sufficiently extensive* way.

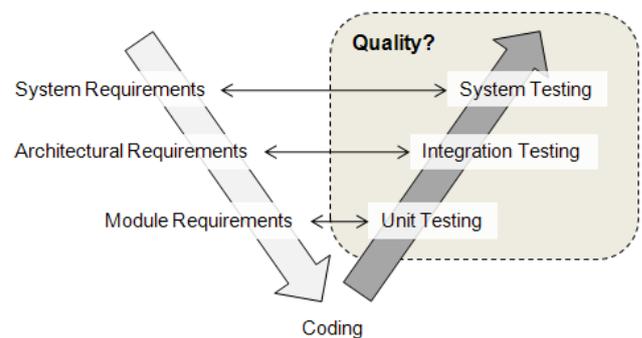


Figure 1 The V process model

Several different coverage metrics exist [7], usually differing on the minimal number of tests necessary to reach full coverage. The most common are statement and decision coverage: the first measures which source code statements are exercised, while the second measures how Boolean expressions (decisions) are evaluated. Reaching a certain degree of coverage means that the requirement-driven testing strategy involves a set of tests whose execution exercises all the structures the metric targets. For example, reaching full statement coverage means executing at least once all statements present in the application source code; reaching full decision coverage means that the tests execution caused all decisions (typically in if, case or while statements) to be evaluated to *both* outcomes (True and False). To provide a practical evidence of the difference between different testing metrics, we introduce here a very simple example which we will use extensively in the remainder of this paper. The source code of the example is the following:

```

1. procedure P (A, B, C : Boolean) is
2. begin
3.   if (A and then B) or else C then
4.     Do_Something;
5.   end if;
6. end P;
```

Listing 1 Example Source Code

The example comprises two statements (lines 3 to 4) and a single decision (the Boolean expression at line 3). Statement coverage for procedure P can be reached with just one test causing the Boolean expression "(A and then

B) or else C" to be evaluated to True. Decision coverage can be reached with two tests, one evaluating the decision at line 3 to True, the other to False.

Coverage analysis is explicitly required by several industrial standards: for example, DO-178B (civil avionics), ECSS-40 (space systems) or IEC-880 (nuclear). In this paper we concentrate on the DO-178B standard because it requires the most stringent coverage metrics to be applied.

2 Structural Coverage and DO-178B

DO-178 [1] is an international standard providing guidance for the development of software to be deployed on airborne systems flying over civil ground. The standard applies to both Europe and North and South America. Developers of avionics software are expected to follow the guidelines contained in DO-178. They shall then submit the required evidence to certification authorities to gain certification credit for the software and allow its deployment on airborne systems. The DO-178 standard has been reviewed in 1992, giving birth to DO-178B. The next revision of the standard, DO-178C, is expected by the end of 2010: the concepts contained in this paper apply to both DO-178B and DO-178C. DO-178B is complemented by DO-248B [2], which provides clarifications on the use of the DO-178B standard.

DO-178B describes a set of development *and* verification activities which shall be performed by a development team in order to gain certification credit for the developed software. Among these activities, structural coverage is part of the software verification process (see [1], table 7). Within DO-178B, structural coverage serves both to assess the quality of the verification strategy and to demonstrate the absence of *unintended functions*: if a part of the application is not covered by a requirement-driven testing strategy, it is likely that the implementation did not proceed from the requirements. Unintended functions shall not be deployed on the final applications, as they are not justified by any requirements, nor they are likely to have been thoughtfully tested.

Three different metrics are considered within DO-178B: Statement Coverage (SC), Decision Coverage (DC) and Modified Condition/Decision Coverage (MC/DC). All of these metrics express coverage in terms of source code elements: notions such as statement, decision, condition are defined *only* in relation to source code. We have already introduced SC and DC. We briefly illustrate MC/DC here: more information can be found on [5,6]. MC/DC distinguishes between *decisions* and *conditions*; a decision is, exactly like in DC, a Boolean expression. a condition is an atomic Boolean expression: conditions are grouped in decisions¹. In the example in listing 1, "A", "B" and "C" are conditions grouped in the decision "(A and then B) or else C". To achieve MC/DC, every point of entry and exit in the

program must be invoked at least once, every decision must take all possible outcomes at least once, every condition in a decision must take all possible outcomes at least once, and each condition in a decision should be shown to independently affect the outcome of that decision [1]. Two evaluation vectors are required to demonstrate independent effect of a condition *C* on a decision *D*. The vectors shall be identical but for the value of *C* and one vector must cause *D* to be evaluated to True, the other to False. For example, to reach MC/DC for the source code in listing 1, we can use the following evaluation vectors for the single decision: (T,T,F), (T,F,T), (T,F,F), (F,T,F). Independent effect of "A" is demonstrated by (T,T,F) and (F,T,F): the two vectors are identical but for the value of "A" and the decision is evaluated first to True, then to False. Independent influence of "B" is demonstrated by (T,T,F) and (T,F,F); independent influence of "C" is demonstrated by (T,F,T) and (T,F,F). MC/DC requires at least n+1 tests to cover a decision composed by n independent conditions [5].

Within DO-178B, the metric chosen for structural coverage depends on the criticality of the application: the more safety-critical the application, the more stringent the metric (i.e. more tests are required to achieve the coverage objective). The aim is to require a more extensive testing campaign for the most safety-critical software applications. The effort required to reach a given coverage shall however be reasonable. For example, path coverage (the execution of *all* possible execution paths) exercises the application more than MC/DC, but it also requires an unreasonable amount of tests. The correspondence between coverage metrics and criticality level is as follows:

- **Level A:** MC/DC. Applications whose criticality level is A are those whose failure would cause a catastrophic impact such as to *"prevent continued safe flight and landing"*.
- **Level B:** Decision Coverage. Applications whose criticality level is B are those whose failure would cause a severe impact such as *"a large reduction in safety margins or functional capabilities, or physical distress or higher workload such that the flight crew could not be relied on to perform their tasks accurately or completely, or adverse effects on occupants including serious or potentially fatal injuries to a small number of those occupants."*
- **Level C:** Statement Coverage. Applications whose criticality level is C are those whose failure would cause major impact such as *"a significant reduction in safety margins or functional capabilities, a significant increase in crew workload or in conditions impairing crew efficiency, or discomfort to occupants, possibly including injuries."*

Level D (minor impact) and E (no impact) do not require any measure of coverage.

2.1 Source Coverage versus Object Coverage

Applicants to DO-178B certification have proposed the use of object code coverage *instead of* source code coverage as a metric to satisfy the objectives of DO-178B. The

¹ A decision may be composed by a single atomic Boolean expression: in this case the condition is also a decision.

proposed approach consisted in measuring either instruction coverage or branch coverage. Object instruction coverage (OIC) requires assessing all object instructions are executed at least once; branch coverage (OBC) in addition requires all conditional branches to be exercised for both directions (branch and fall through). The use of object code coverage has also been proposed as a way to cope with *untraceable object code*. Section 6.4.4.2 of DO-178B indeed states: *"The structural coverage analysis may be performed on the Source Code, unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences"*. Untraceable object code is object code generated by the compiler and impacting the control flow in a way not directly visible from source code. A typical example is array bounds check which may raise an exception. Reaching a given level of coverage for source elements may not be *representative* of the untraceable object code: even an extremely complete testing campaign may not assure all object code is executed. The untraceable object code is nevertheless present in the application, it may be executed during operation and may thus lead to unintended behaviour not verified during the testing process. Measuring object code coverage may thus be a way to ensure even untraceable code is executed during the requirement-driven testing campaign.

The issue raised by section 6.4.4.2 of DO-178B, and in general the equivalence of source and object coverage, is considered in both FAQ 42 of DO-248B (*"Can structural coverage be demonstrated by analyzing the object code instead of the source code?"*) and CAST paper 17 (*Structural Coverage of Object Code*, [3]) issued by the FAA (Federal Aviation Administration). Both documents assert that object code coverage can substitute source code coverage *"as long as analysis can be provided which demonstrates that the coverage analysis conducted at the object code will be equivalent to the same coverage analysis at the source code level"* [2]. In this context, equivalence means that object code coverage should require the same number of test cases as those needed to reach source code coverage for the appropriate metric (CAST paper 17 at [3]). In practice, object code coverage may be equivalent to source code coverage only on limited cases, for example when [4,12]: (i) the compiler generates a branch instructions for each condition and (ii) only short circuit operators (and then, or else) are used and (iii) short circuit Boolean operators are always right-associated, like **"A and then (B or else C)"**. When Boolean operators are right associated, the nodes of the resulting evaluation graph always have a single incoming edge (coming from the previous condition) and two outgoing edges: one exits the evaluation, the other proceeds to the next condition. The evaluation graph is thus a tree with $n+1$ leaves: there are $n+1$ paths from the root to the leaves, meaning that we need $n+1$ test vectors to fully cover the tree and achieve full OBC, exactly like for MC/DC (fig. 2a). When Boolean operators are left-associated (like in listing 1), the resulting

evaluation graph is not a tree and can be potentially covered by less than $n+1$ tests (see fig. 2b). In the general case, it is rarely the case for object code coverage metrics to be in any direct relation to source code coverage metrics. For example, branch coverage is not equivalent to MC/DC, and instruction coverage is not equivalent to DC or SC. FAA reports such as DOT/FAA/AR-07/17 [4] provide evidence of this absence of relationship. It is also pretty simple to provide an example using the code in Listing 1. When compiling the code in listing 1 with GNAT Pro 6.2.1 for a bareboard PowerPC platform and with `-O1` optimization, the generated object code is (branch instructions in bold):

```
_ada_p:
  stwu 1,-16(1)
  mflr 0
  stw 0,20(1)
  cmpwi 7,3,0      # compare r3 (A) with 0 and put result in cr7
beq- 7,,L2      # if equal (A=False) branch to L2
  cmpwi 7,4,0      # compare r4 (B) with 0 and put result in cr7
bne- 7,,L3      # if not equal (B=True) branch to L3
.L2:
  cmpwi 7,5,0      # compare r5 (C) with 0 and put result in cr7
beq- 7,,L5      # if equal (C=False) branch to L5 (end)
.L3:
  bl do_something_pkg__do_something
```

The resulting execution graph is shown in figure 2b. Note that the same graph would be produced even with no optimization (`-O0`). Full branch coverage can be reached with just three tests (using test vectors (T,T,F), (T,F,T), (F,T,F)), while MC/DC requires at least four tests. The basic reason for such a difference is MC/DC begin a stateful property over the evaluation of a decision, whereas OBC is a local property of a single instruction.

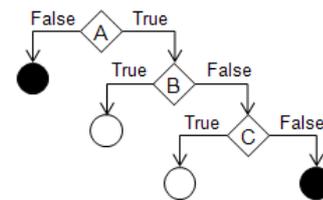


Figure 2a Evaluation graph for "A and then (B or else C)"
Four tests are necessary to cover the whole graph.

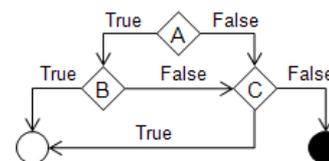


Figure 2b Evaluation graph for "(A and then B) or else C":
Branch coverage achieved with (T,T,F), (T,F,T), (F,T,F)

Object code coverage is also just a *partial* solution to cope with the additional verification activities on untraceable object code for level A software (DO-178B, section

6.4.4.2). Producing additional tests to cover all untraceable object code has little sense because the requirements that caused such code to be produced cannot be found in the specifications of the system under development, but rather in the way the compiler implements a part of the programming language standard. Let's consider again array bounds check: they are generated because this is how the compiler implements the language standard, not because of some requirement specific to the application being developed. Moreover, covering all *untraceable* object code may require a significant effort which would be better spent in performing more meaningful verification activities. This is why CAST paper 12 [3] suggests the use of a *traceability study* to satisfy the additional verification activities on untraceable object code for level A software: a traceability study provides evidence that, in a given context (coding standard, compiler, compilation switches), the compiler either generates traceable code or the untraceable code is correct – i.e. it correctly implements the requirements expressed by the specifications of the chosen programming language. A traceability study is thus a sort of black box qualification of the compiler: the requirements are the parts of the language standard used in the application context and tests compare the object code generated by the compiler with the legitimate expectations.

Regardless of guidance and empirical evidence, the equivalence of object code coverage and source code coverage is however still a controversial topic and subject of debate: applicants managed to gain certification credit using both source code coverage and object code coverage.

3 Current approaches to structural coverage

Current industrial solutions to evaluate structural coverage are usually divided into two main categories: tools measuring source code coverage and tools measuring object code coverage. Given the domain of interest (aerospace), it is important to remind that the source code is usually cross compiled on host workstations and deployed on less powerful target platforms.

Tools measuring source code coverage like IBM Rational Test RealTime, LDRA Testbed, IPL AdaTest or Bullseye Coverage, usually depend on source code *instrumentation* to evaluate the coverage. Source code instrumentation requires the coverage tool to *modify* the application source code by adding calls to logging facilities in appropriate locations. For example, to measure statement coverage it is enough to log the entry in each block of execution (if statements, loops, switch statements, etc.): if all blocks are entered and no exception has been raised, then all statements have been covered. Of course, the more stringent the coverage metrics, the more invasive the instrumentation: instrumenting the source code for MC/DC requires to log the evaluation of each atomic Boolean expression, as well as to record all Boolean evaluation vectors to demonstrate independent effect of conditions on decisions. Coverage of instrumented code has the advantage of providing a straightforward mapping between the source code and the logged coverage information: it

perfectly fits the requirements of DO-178B. On the other side, developers may be asked to demonstrate that the instrumentation did not modify the application behaviour and that the coverage of the instrumented application is *representative* of the coverage of the final application. It is usually necessary to compare the tests results for the two versions of the application (instrumented and not instrumented) to provide such evidence.

The other approach to structural coverage targets object code coverage. Tools measuring object code coverage usually log which instructions have been executed by exploiting hardware probes connected to the target via a JTAG/BDM, debug information and instruction-by-instruction execution. This approach has the major advantage of being *non intrusive*: since no instrumentation is added, and since coverage is measured on the final, cross compiled application, no additional verification is required. In addition, this approach depends on the target platform rather than on the programming language, making it an excellent candidate for applications written with several programming languages. The main limitation consists in the dependence upon the target hardware (and a physical connection) to execute the tests and measure coverage: this means that coverage may be measurable only when the target hardware is available; furthermore, such a process suffers from the slowness of the hardware connection to the target hardware required by the technology. Regardless of the limitations described in section 2.1, several successful tools measure object code coverage, for example VeroCel VeroCode or GreenHills GCover.

In this section, we identified the following major limitations in current state-of-the-art coverage approaches:

For approaches based on source coverage via instrumentation: it is necessary to demonstrate that instrumentation did not modify the behaviour and coverage of the final, non-instrumented application.

For approaches based on object code coverage: coverage results cannot be directly mapped onto source coverage metrics and they usually require the final hardware to be available and connected to the host via a probe.

4 Couverture

Couverture is a research project founded by French institutions within the System@tic framework. The project consortium comprises AdaCore, OpenWide, Telecom ParisTech and LIP6 (Pierre et Marie Curie University, Paris).

Couverture innovates on all aspects of current technology for structural coverage by:

- Providing a virtualized *execution platform* for cross-compiled application on the host machine. The virtual machine is able to produce a detailed *execution trace*.
- Measuring *object code coverage* through careful examination of execution traces.
- Measuring *source code coverage* as defined by DO-178B by relating elements of the execution trace

(instructions, branches) to source-level structures (statements, decisions, conditions).

The Couverture technology is thus able to measure *both* source and object coverage for the *cross compiled* application. Supporting both object and source code coverage guarantees the maximum flexibility in terms of user needs. The use of a virtualized environment guarantees an extremely efficient process because it does not require the hardware to be available nor to be physically connected to the target board. Finally, since source code coverage is inferred from execution traces, no instrumentation of source code is required, making it unnecessary to provide evidence that the natively compiled, instrumented application is equivalent to the cross compiled one.

In the following sections we examine in details the main point of interests of Couverture.

4.1 A virtualized, instrumented execution environment

The core of the Couverture technology is a virtualized execution environment playing a dual role: it permits the execution of cross compiled applications on the host workstation without requiring the final hardware to be available and it gathers execution traces exploited by Couverture to measure object and source code coverage. The basic idea behind Couverture is to virtualize the approach commonly used to measure object code coverage: while traditional object coverage tools require a physical connection to the target hardware to measure coverage, Couverture uses a virtualized environment producing a rich execution trace containing the address of executed object instructions and the outcomes of conditional branches. Couverture is then able to determine actual object code coverage by *processing* several execution traces corresponding to the executions of different tests.

The technology at the heart of Couverture is QEMU (<http://www.qemu.org>). QEMU is a processor emulator employing dynamic binary translation. QEMU takes as input a cross-compiled application and translates basic blocks into executable code for the host processor. Thanks to binary translation, and to the fact that the target platform is usually much slower than the host workstation, performances are more than adequate: in the case of current generation PowerPC or ERC32/LEON2 targets, the simulator proved to be faster than the target boards. QEMU has an accurate model of the target Floating Point Unit (FPU). Although using an emulated FPU is slower than directly accessing the host FPU, the emulator assures a numerical precision identical to the target: from a numerical computations point of view, QEMU is *representative* of the target platform. QEMU also emulates different I/O chips which can be used to simulate the interaction between the application and the environment. Thanks to the emulation of I/O, QEMU permits to run also those testing campaigns which would require a *physical* interaction with the external environment. QEMU is released as *free* software: the availability of the source code and the presence of an active user community guarantee the

tool can be extended to support additional hardware and logging techniques. Our main extension to QEMU is the support for the generation of execution traces containing the list of executed object instructions: our addition required minimizing the amount of logged instructions to avoid the explosion of the size of execution traces. By comparing this list with the dump of the executable object, it is possible to measure object code coverage. In addition, the virtualized environment is able to keep track of the history of the evaluations of branch instructions: each single evaluation of a branch instructions is tracked along with the value to which the branch instruction was evaluated (branch or fallthrough). This additional information is necessary to measure branch coverage because passing through a branch instruction just once is not enough to achieve its full coverage.

Currently, the following platforms are supported via a QEMU-based emulation: PowerPC bareboard, LEON2 and ERC32 (SPARC) bareboard and WindRiver VxWorks 653 on PowerPC.

4.2 Introducing Source Coverage Obligations

The augmented execution traces produced by QEMU are enough to measure object code coverage. They are however not sufficient to infer source code coverage: without additional data, Couverture would suffer the same limitations plaguing other coverage tools targeting object code coverage (see section 2.2). Couverture needs to relate source code structures to object code elements and in particular to determine:

1. Which object code *instructions* are generated from each statement;
2. Which *statement* (and thus which instructions, see point 1) is executed if a *decision* is evaluated to True and which is executed if it is evaluated to False;
3. Which *branch instruction* corresponds to which *condition*;
4. How *conditions* (and thus branch instructions, see point 3) are grouped together to form a *decision*.

Standard debug data does not provide all required information. Debug formats such as DWARF 2/3 can just link object instructions to a location in the source code identified by a line and a column, but are not able to map them to source-level *structures* such as conditions or decisions. This is why we had to *complement* the debug information with *Source Coverage Obligations (SCOs)*. SCOs are extracted from source code by the compiler: they identify source constructs for which coverage artifacts need to be exhibited in order to satisfy some coverage objective. SCOs contain a compact representation of the control flow which group conditions within the enclosing decision and specify which statements are executed when a decision is evaluated to True and False. By weaving SCOs with debug information, it is possible to represent the source-level control flow and its structures in terms of object instructions. Couverture can then infer all DO-178B source

code coverage metrics from the execution traces produced by QEMU:

- **Statement coverage** is achieved if any instruction generated by a given statement are executed (see point 1 above).
- **Decision coverage** is achieved if the Boolean expression composed by all object branch instructions associated to a given condition is evaluated to both True and False. Calculating decision coverage requires a sort of abstract interpretation of execution traces to determine the evaluation of a decision starting from the evaluation of the associated object branches.
- **MC/DC** is achieved if all branch instructions have demonstrated to have an independent impact on the evaluation of the decision they belong to (see point 4 above). Of course, this includes each branch instructions to be evaluated for both directions (branch and fallthrough, see point 3 above)

As explained, SCOs *and* debug information contain the information to detect whether the conditions above apply. An example shall help us clarifying this crucial point. Starting from the decision at line 3 of the procedure P in listing 1, the GNAT Pro compiler is able to produce the data depicted in fig. 3: it associates conditions to branches and it groups branches corresponding to the evaluation of a given decision. Thanks to this information, Couverture is able to detect that the three tests sufficient to achieve branch coverage (see fig. 2) are not enough for MC/DC: since the SCO states that all three branch instructions are *logically* owned by the same condition, Couverture can easily verify that the independent impact of decision "C" on the whole decision is not demonstrated (test vector (T, F, F) is missing). This example shows that the coverage technology employed by Couverture does not suffer from the same limitations of other object code coverage tools.

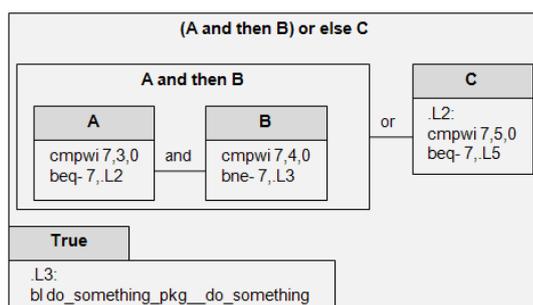


Figure 3 Simplified graphical representation of a SCO plus debug information produced by GNAT Pro

4.3 Preserving execution flow

The mechanism employed by Couverture to measure code coverage consists in inferring source code execution flows by analyzing object code execution traces with the support of SCOs: for this to be possible, the control flow across object instructions must be representative of the control flow across source code elements. In particular, each condition must be mapped onto a branch instruction. It is

thus extremely important to preserve the execution flow from source code to object code: if control structures are not preserved during compilation, it may be impossible to infer paths across source code elements from object code execution traces. A simple example will help clarifying this idea. Consider the following function:

```

1. function F (A : Boolean) return Boolean is
2. begin
3.   if A then
4.     return True;
5.   else
6.     return False;
7.   end if;
8. end F;
```

Listing 2 Example Source Code

When compiled with minimal optimization (-O1), the resulting object code is:

```

_ada_f:
blr    # return the value of the parameter
```

The generated object code does not contain any branch instruction: the execution flow implemented by the object code is not representative of the execution flow in the source code. As a result, even the Couverture toolset would assert that *any* form of DO-178B coverage can be achieved with just one test, instead of the two actually necessary: in fact, a simple execution of function F will cause the *complete* coverage of all generated object code. In the general case, even disabling optimization cannot guarantee the source code execution flow is preserved down to object code. To address this issue, the compilation technology shall guarantee the execution flow is preserved, even when using optimizations. In GNAT Pro (AdaCore flagship Ada compiler), this is achieved by the `-fpreserve-control-flow` switch. `-fpreserve-control-flow` disables all requested optimizations (possibly via the `-O` option) which may influence the preservation of the control flow from source code to object code. The result of using this compilation switch with the source code in listing 2 is the following:

```

_ada_f:
cmpwi 7,3,0
beqlr- 7    # branch instruction for A
li 3,1
blr
```

In the example, a branch instruction is generated for the condition "A" in the source code: this ensures the control flow described at source level is preserved in object code instructions.

5 Current Results

The Couverture technology is currently used both as an internal tool and within an industrial context. In both cases, Couverture has been coupled with a GNAT Pro compilation chain.

5.1 Industrial Test Cases

Couverture is currently used at AdaCore to measure the source code coverage of the internal test suites used to

qualify the Ravenscar run-time for the ERC32 and LEON2 processor in a ECSS-40 context. QEMU has been extended to support these processors.

In addition, the development team responsible for the Air Data Inertial Reference Unit (ADIRU) for the Airbus A350 XWB at Thales Aerospace uses Couverture to measure object *and* source code coverage at DO-178B level A [10]: in this context, QEMU emulates a bare-board PowerPC target.

5.2 R&D Test Cases

Couverture has also been evaluated using applications developed within two R&D projects. The first test case is represented by an Ada 2005 application built in the context of the IST-ASSERT project. The application is a reduced subset of the guidance and navigation system of a satellite. Coverage analysis with Couverture uses the support for LEON2 in QEMU. We exercised the application in nominal mode, and processing the output with the Couverture toolchain. From the analysis, we found out that 71% percent of the code was fully covered, whereas 25% was partially covered. Only 4% was not covered. The coverage analysis revealed that the partially covered code corresponded to error conditions not being exercised, and thus showed that our test suites didn't completely cover our requirements.

The second test case resolves around POK, a real-time embedded kernel for safety-critical systems. POK implements several industrial standards, including ARINC653 used in the avionic domain. POK comprises two main layers: the kernel and the partition runtime (called libpok). The kernel provides partition support, inter-partition communication and scheduling; it is particularly compact in order to (i) ease its verification/certification and (ii) minimize the amount of potentially unintended functions. The libpok layer provides kernel-interfacing functions and abstraction layers, for example, POSIX or ARINC653. Partitions are executed on top of the POK kernel to ensure partitions isolation. Each partition can include one or more compatibility layers to execute their application code. Each partition is also isolated in space (a memory segment is dedicated to each partition) and time (each partition has at least one time slot to execute its threads). POK is written in C for x86 and PowerPC architectures. Ongoing work includes a LEON port and an Ada API for the ARINC653 layer of libpok. POK is available as open-source software under the BSD licence at <http://pok.gunnm.org>.

Since the libpok layer contains several heterogeneous components, its full inclusion would likely lead to deploy code not derived from requirements in the final application. To avoid the deployment of unintended functions, kernel and libpok can be configured to explicitly select the required services using an automatic code generator from AADL specifications [13]. The code generator configures each layer according to system requirements and thus ensures configuration correctness and absence of unintended functions.

We have carried out experiments to analyze kernel code coverage using Couverture (statements coverage). We studied two examples of POK using different kernel services:

- “Partition-threads”: one partition that contains two tasks. It only uses the partitioning services.
- “Middleware-queuing”: two partitions with one task per partition. Inter-partition communication occurs between the two partitions so the kernel provides partitioning and inter-partition communication services.

The measured code coverage for x86 targets ranges from 65% (middleware queuing) to 91% (partition threads). This is mostly expected: more complex setup in the second example leads to potentially more dead code. Thanks to Couverture we were thus able to identify which services of the POK kernel should be moved to libpok in order to include them only when strictly necessary: after these changes, POK and its test cases shall comply with the coverage requirements for DO-178B level C. In addition, we managed to improve our code generation strategies to decrease the amount of dead code coming from non-necessary libpok services. Both case studies demonstrate that Couverture is ready to address the challenges of certification of systems deployed on the LEON2 processors and of applications executed on partitioned kernels.

6 Conclusions

In this paper, we have illustrated an innovative approach to structural coverage based on a virtualized, instrumented, execution environment. Couverture is able to measure structural coverage of object *and* source code *without* requiring any form of application instrumentation with a *single* execution of the *cross-compiled* application and test suites. To measure source code coverage, the Couverture technology requires the compiler to identify source constructs for which coverage artifacts need to be exhibited and to link them to object instructions. This is achieved by the generation of Source Coverage Obligations and debug information. The compiler must also assure the execution flow is preserved from source code to object code. The GNAT Pro compiler has been extended to implement such feature via the `-fpreserve-control-flow` switch. The technology described in this paper is currently used both as an internal tool and in a major project in a DO-178B level A context.

In addition to its technical contributions, Couverture is the first industrial project which leverages on the Open-DO vision [9]. Open-DO is an initiative to promote an open and collaborative approach to the development of high integrity systems, in particular within a DO-178 context. Couverture completely embraces the Open-DO vision, providing an open repository [11] where development artifacts are freely available: potentially, the whole user community can access and *contribute* to the development and qualification of the technology. This is the first step towards the cross fertilization between *open* and *high-assurance* development promoted by Open-DO.

References

- [1] EUROCAE: "Software Considerations in Airborne Systems and Equipment Certification" - DO-178B, 1992, 1999.
- [2] EUROCAE: "Final Report of Clarification of DO-178B" – DO-248B, 2001.
- [3] FAA CAST papers, http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/
- [4] FAA: "Object-Oriented Technology Verification Phase 3 Report - Structural Coverage at the Source-Code and Object-Code - DOT/FAA/AR-07/20", 2007
- [5] FAA: "An Investigation of Three Forms of the Modified Condition Decision Coverage (MCDC) Criterion - DOT/FAA/AR-01/18", 2001
- [6] NASA: "A Practical Tutorial on Modified Condition/Decision Coverage - NASA/TM-2001-210876", 2001
- [7] Ntafos, S.C: "A comparison of some structural testing strategies", IEEE Transactions on Software Engineering, Issue 6, 1988
- [8] Beizer, B., "Software Testing Techniques", 2nd edition
- [9] Open-DO: www.open-do.org
- [10] Thales Aerospace Division Selects GNAT Pro for Airbus A350 XWB (Xtra Wide-Body): <http://www.adacore.com/2009/06/01/a350/>
- [11] Couverture Open Repository at Open-DO.org: <http://forge.open-do.org/projects/couverture>
- [12] Romanski G.: "MCDC coverage using short0circuit conditions", available at verocel.com.
- [13] Delange J., Pautet L., and Kordon F.: "Code Generation Strategies for Partitioned Systems", 29th IEEE Real-Time Systems Symposium (RTSS'08), work in progress proceedings, December 2008

Generating Component-based AADL Applications with MyCCM-HI and Ocarina

Thomas Vergnaud, Frédéric Gilliers, Hugues Balp

THALES Communications, 1, avenue Augustin Fresnel, Palaiseau, France; firstname.lastname@thalesgroup.com

Abstract

This paper presents the result of studies made on the combination of Lightweight CCM and AADL in order to build component-oriented applications that meet requirements for real-time embedded systems. We explain how we mapped Lightweight CCM components onto AADL architectures, and the consequences of such a combination. We finally present some results in terms of memory size and execution jitter. This work was done in the scope of two projects: ANR Flex-eWare and ITEA SPICES.

Keywords: Lightweight CCM, AADL, distributed real-time embedded systems (DRE), model driven engineering (MDE).

1 Introduction

Application design for real-time embedded systems traditionally relies on what we could call “low-level technologies”, such as direct Ada programming. Approaches like model driven engineering (MDE) are seldom used, since they tend to deal with more abstract views of the applications, and thus often prevent the designer from controlling all aspects of the actual application code.

Components oriented approaches, which focus on the application logical topology, are known to help capture the complexity of distributed embedded systems [5]. When available, automatic code generation associated to these approaches greatly improves development efficiency and helps reduce the cost of complex systems. However the component based approaches have mainly been adopted in information systems [7], where time constraints are less stringent and computing resources much less sparse.

Research has been done to study how to describe real-time embedded architecture [4], notably using architecture description languages such as AADL [1]. However, AADL remains at a low level of description, and thus does not focus on component-based software architectures.

In this paper, we explain how we combined these two approaches in order to provide a high-level, business-oriented application design while ensuring real-time embedded capabilities for the generated application code. In the first section, we briefly describe the two standards we used: Lightweight CCM and AADL, and the relationship we can define between them. Then we provide some details on the mapping we defined from one to the other. Finally we present some results in terms of

implementation with execution jitters and memory footprint to validate our approach.

2 Overview of Lightweight CCM and AADL

In this section, we describe the main aspects of Lightweight CCM and AADL, and we explain how they can be associated in a common process.

2.1 Lightweight CCM

Lightweight CCM [3] is a standard subset of the CCM (CORBA Component Model) standard component model. It is defined by the OMG. Lightweight CCM is dedicated to providing a CCM compatible component model suitable to the needs of distributed embedded systems.

Lightweight CCM defines the notion of software component as an envelope that wraps the user business code, isolating it from the execution environment. User code thus communicates with the outside of the component only through the component envelope. This envelope is described using the IDL3 language. IDL3 defines two communications ways offered to the user code: interfaces and events. Interfaces are sets of operations; they can be either provided by a component (facets) or required by it (receptacles). The same way, events are either sent by the component (event sources) or received by it (event sinks).

IDL3 constructions are transformed into sets of IDL2 interfaces that are to be implemented either by the component framework (i.e. the component envelope) or by the user code. This later case corresponds to the services provided by the component, described in its IDL3 declaration. This set of interfaces is named CIF (Component Implementation Framework).

CCM is originally defined as the CORBA 3 standard, and thus typically relies on an ORB to manage communications. However, this is not mandatory, as the CCM purpose is to hide the ORB from the business code, nested in components. Therefore, one can use virtually any runtime to support the execution of CCM architectures, provided that it can manage the two communications paradigms (operations and events).

IDL3 itself does not address the description of component deployment. The CCM is thus usually associated with another OMG standard, D&C [2] that covers the deployment and the configuration of components. D&C is a very rich and complex standard, mainly adapted to the deployment of complex, dynamic information services. It lacks several configuration elements required to deploy

real-time systems (e.g. thread priority definition). Therefore, in the context of this work, we developed a custom architecture language named COAL (Component Oriented Architecture Language). COAL [8] is targeted at the description of static deployment of IDL3 components.

2.2 AADL

AADL is an architecture description language targeted at the modeling of real-time embedded systems, distributed or not. It is standardized by the SAE [1].

Compared with Lightweight CCM, AADL covers wider modeling aspects: it allows the description of software architectures in a hardware execution environment. Software elements are described from a low-level point of view, centered on execution threads.

AADL defines a notion of components, which do not have the same semantics as in Lightweight CCM. Components have an interface that provides features and implementations that describe their internal structure.

The AADL standard defines several categories of components; each of them has well identified semantics. Software components are used to model application. *Processes* represent memory spaces in which *threads* are executed. Processes have to contain at least one thread. *Thread groups* are used to gather threads and thus create thread hierarchies. *Subprograms* correspond to programming language procedures. *Data* are used to model the data structures manipulated by AADL threads and subprograms.

Execution platform components are used to describe the hardware environment topology on which applications are deployed. *Processors* represent microprocessors and associated operating systems. *Memories* represent storage units, such as RAM, hard disks, etc. *Devices* are used to model sensors or purely hardware components. *Buses* represent networks, wires, etc. that are used to interconnect processors, devices and memories.

AADL *systems* are used as containers for other components (including systems); they help in structuring architectures. *Abstract* components are also defined; they carry no semantics at all, and are simple boxes that can be refined in any other component category.

Components can contain subcomponents. The standard defines legality rules regarding the possible compositions: processors cannot be subcomponents of subprograms; threads can (and must) be subcomponents of processes.

Component features can be *ports*, *required* or *provided access* to some subcomponents or subprograms. Inside components, features of subcomponents are connected to features of other subcomponents or to features of the containing component. Features of threads are used to model different kinds of communications paradigms: message passing (event data ports), remote operation call (subprogram accesses), and shared memory (data access).

Component composition describes the structure of architectures. AADL also allows for the characterization of

these components: properties can be associated with each architecture element (components, subcomponents, features, connections, etc.).

AADL properties are used to specify constraints (e.g. execution time, memory size) or characteristics (e.g. period, thread dispatch policy). They can also be used to describe configuration parameters (e.g. the network address of a processor). Properties can also be used to associate source code files to application components; AADL components act as containers for the application algorithms.

The AADL defines sets of standard properties, for which precise semantics is defined. Thus, analysis and generation tools can interpret them. Users can also define their own properties. The use of these additional properties is then dependent on specific tool support, e.g. scheduling or reliability analysis.

AADL constructions describe application structures that encapsulate algorithms. Hence, such structures control the execution of the application, according to the different parameters provided by the architecture description (topology and execution parameters given by AADL properties).

An AADL application is executed on the top of an AADL runtime that provides scheduling and communication services. Such a runtime is configured according to the architecture description. Standard AADL properties are interpreted to parameterize some aspects of the runtime (e.g. execution periods, dispatch policies, etc.)

3 Rationale for combining modeling languages

Lightweight CCM (i.e. IDL3) and AADL are two distinct modeling languages. Yet, they address different modeling aspects: Lightweight CCM focuses on the logical software description, without dealing with deployment and resource allocation. AADL focuses on resource allocation and precise runtime description, but does not integrate software business components (in the sense of Lightweight CCM). Lightweight CCM describes architecture from a more abstract point of view than AADL. Therefore, combining these two approaches provides a complete model-based design process with support for effective generation.

Mappings between modeling languages consist in defining equivalences of concepts between them. Most of the semantic aspects must be preserved in order to have a meaningful mapping. In our situation IDL3 and AADL address different modeling scopes. IDL3 covers business component modeling. AADL covers deployment and resource allocation. Hence, we cannot define a mapping that would preserve all semantics.

The mapping from IDL3 to AADL actually defines how to build an AADL application with AADL semantics from an IDL3 design. The following section described the mapping we defined.

4 Mapping of Component Declarations

The mapping from Lightweight CCM to AADL is structured into several parts:

- Units (i.e. modules, or packages);
- Data types;
- Component types;
- Resource allocation and deployment.

4.1 Modeling Units

Model units are packages in AADL and modules in IDL3. Packages and modules help structure the declarations.

IDL3:

```
module a_unit {
// public declarations
...
// no private declarations
};
```

AADL:

```
package a_unit
public
-- public declarations
...
end a_unit;
```

IDL3 is a language to describe interfaces. Thus IDL3 modules have no private section, since it would have no sense. On the contrary, AADL is used to describe the complete structures of the architectures, including components inaccessible from external entities. Therefore AADL packages can have a private section, which is not used by the mapping.

4.2 Data Types

IDL3 defines a set of data types that can be manipulated by the components. AADL does not have exactly the same approach. The definition of the data semantics is not part of the core standard. Thus, AADL does not impose any specific notation to associate semantics to data components.

An annex of the AADL standard defines a set of properties to specify the semantics of data components [6]. It offers more flexibility than IDL3. Therefore, most IDL3 data types can be described in AADL. Since this property set is not normative, one can use its own way of definition as well.

In the scope of a mapping between IDL3 and AADL, it is sound to directly associate the type definitions written in IDL3 with AADL data component declarations.

Basic types

In the scope of our architectures, basic IDL types with a bounded size (short, unsigned short, long, unsigned long, long long, unsigned long long, float, double, long double, characters, wide characters, bounded strings, bounded wide strings, boolean, octets) are allowed.

Unbounded strings should not be allowed, since their use prevents from statically computing buffer sizes during the code generation process. Such a computing is required for static allocation strategies required by real-time systems. Equivalent AADL data component declarations are defined in an AADL package named CORBA (as the IDL3 types are standard CORBA types). These data components redefine standard AADL data components.

```
package CORBA
public
data short extends Base_Types::int16
end short;
data unsigned_short extends Base_Types::uint16;
end unsigned_short;
...
end CORBA;
```

Complex types

All complex IDL types whose size can be computed at compilation time are allowed: enumerations, fixed-point numbers, structures, union, fixed-size arrays, and bounded sequences.

Unbounded sequences and CORBA Any types are not allowed, since it is not possible to statically determine their size..

The following syntactical IDL constructions are managed:

```
typedef <type> a_typedef;
struct a_struct_type {...};
enum an_enum_type {a, b};
union a_union_type switch (<discriminator>) {...};
typedef <type> an_array_type [<dimension>];
sequence <type, max_elements> a_sequence_type;
```

They are translated into the following AADL declarations:

```
data a_typedef extends <type>
end a_typedef;
data an_enum_type
properties
data_model::data_type => enum;
data_model::enumerators => ("a", "b");
end an_enum_type;
data an_array_type
properties
data_model::data_type => array;
data_model::base_type => (data <type>);
data_model::dimension => <dimension>;
end an_array_type;
...
```

4.3 Component Types

CCM components are used for business-oriented design, while AADL components correspond to concrete software or platform entities.

Specific semantics is associated with each AADL component categories, except abstract components. On the contrary, CCM components carry no precise semantics concerning their management by the runtime.

System threads that are attached to component interfaces drive CCM components. Hence, CCM components typically correspond to sets of AADL threads and possibly other components (data, subprogram groups...). The exact translation of CCM components into AADL depends on the deployment information. Interfaces described in IDL3 are associated with AADL components.

Plain IDL3 descriptions do not provide any deployment and resource allocation information. In these cases, IDL3 components are to be translated in a first step into AADL abstract components.

IDL3:

```
component a_component {
// interfaces and ports
...
};
AADL:
```

```
abstract a_component
features
-- features
...
end a_component;
```

When adding deployment information, the AADL abstract component shall be replaced by the adequate concrete AADL components (typically, AADL threads). Note that since CCM and AADL components do not correspond to the same modeling scope (business and actual implementation), the mapping between CCM and AADL actually associates a set of IDL3 components with a set of AADL components: AADL components may be “shared” by several CCM components.

Synchronous communications in CCM are performed using operations grouped in interfaces. On the AADL side, feature groups must be used. These feature groups contain subprogram accesses.

IDL3:

```
interface iface1 {
void method_a (in long param);
long method_b (inout short param1,
out boolean param2);
};
interface iface2 {
double method_a ();
};
component a_component {
provides iface1 interface1;
uses iface2 interface2;
};
```

AADL:

```
subprogram iface1_method_a
features
param : in parameter CORBA::long;
end iface1_method_a;
subprogram iface1_method_b
features
```

```
param1 : in out parameter CORBA::short;
param2 : out parameter CORBA::boolean;
result : out parameter CORBA::long;
end iface1_method_b;
subprogram iface2_method_a
features
result : out parameter IDL_types::double;
end iface2_method_a;
feature group iface1
features
method_a : provides subprogram access
iface1_method_a;
method_b : provides subprogram access
iface1_method_b;
end iface1;
feature group iface2
features
method_a : provides subprogram access
iface2_method_a;
end iface2;
abstract a_component
features
interface1 : feature group iface1;
interface2 : inverse of feature group iface2;
end a_component;
```

Methods provided by IDL3 components are translated into accesses to AADL subprograms provided in the AADL component features.

Asynchronous communications in CCM are performed through event sources and event sinks. The equivalent constructions in AADL are in or out event data ports.

IDL3 event types correspond to declarations of AADL data components that are semantically structures.

Unlike operations, CCM event ports are directly declared in components and are not part of any interface.

IDL3:

```
eventtype type_of_communication {
public short a;
public long b;
};
component a_component {
publishes type_of_communication event1;
emits type_of_communication event2;
consumes type_of_communication event2;
};
```

AADL:

```
data type_of_communication
properties
data_model::data_type => struct;
data_model::base_type => (data CORBA::short,
data CORBA::long);
data_model::fields => (“a”, “b”);
end type_of_communication;
component a_component
features
event1 : out event data port type_of_communication;
```

```

event2 : out event data port type_of_communication;
event3 : in event data port type_of_communication;
end a_component;

```

We can note that there is no distinction between publishes and emits in AADL component features: this distinction impacts the number of AADL connections that should be attached to the out ports.

The internals of the abstract component is made of a set of AADL threads that correspond to the resources allocated to the CCM component to manage its interfaces. They encapsulate the user code of the CCM component, as well as the envelope implementation code that connects the CCM API (i.e. the Component Implementation Framework) to the AADL runtime API.

5 Mapping of Deployment Information

In this section, we describe how to translate deployment information associated with CCM descriptions into AADL architectures. We provide a general approach to handle the deployment information that does not rely on any deployment formalism, such as OMG D&C, COAL or even UML/MARTE. The information we consider is:

- software and hardware nodes (i.e. processes and processors);
- binding of CCM components to nodes;
- connections between nodes;
- connections between component interfaces;
- allocation of execution resources to component interfaces.

We actually worked with COAL, but the mapping is not bound to a specific syntax, provided that the necessary information is provided by the description. Therefore, we do not provide examples of COAL concrete syntax in this section.

CCM components that correspond to software implementations are gathered in AADL processes, bound to AADL processors. AADL threads describe the executions resources that are required to run the CCM components.

All CCM components port instances can be associated with an execution resource. If no execution resource is allocated to a CCM port, this port will not appear in the AADL description, and the connection will be managed inside the AADL thread code (generated by the component framework).

As CCM event are considered to be asynchronous communications, an execution resource must be allocated to all CCM event sinks.

Since the dependencies between facets/event sinks and receptacles/event sources are not described in IDL3, all receptacles and event sources of all the CCM components managed by a given AADL thread are translated as features of this AADL thread.

The transformation rules are as follows:

1. The CCM components that are collocated into a given software node must be identified;
2. An AADL processor and an AADL process are created;
3. All ports (facets, receptacles, event ports) that are connected to external components are translated into AADL and associated with the AADL process;
4. An AADL thread is created for each execution resource declared in the component instance deployment description; all the threads corresponding to a given interface are identical;
5. Each AADL thread manages the input CCM interface it is associated with, as well as all output interfaces that may be invoked by the input interface; components providing “passive” interfaces are merged with the calling component; thus, each AADL has the output interfaces of the CCM component they manage, and also the output interfaces of the CCM components that are invoked through “passive” interfaces;
6. The source code associated with the component implementation, as well as the source code that manages the connection between the CIF and the AADL runtime API, are associated with the AADL process using the two standard properties `source_language` and `source_text`;
7. The source code of the envelope that must be invoked is specified using the property `compute_entrypoint_source_text`, associated with each thread.

The following pictures show a deployment example in CCM and the resulting AADL architecture. Components B and C are deployed on a software node, and connected to other components on other nodes.

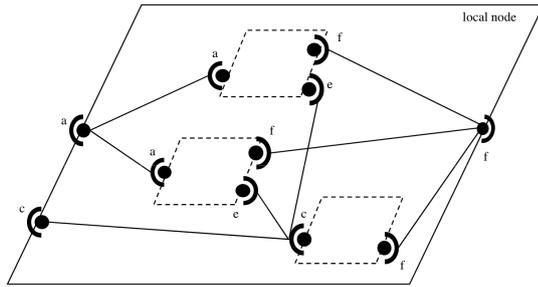
The software node is represented in AADL by a process bound to a processor. Once again, only ports a, c, and f are connected to external components. Therefore, only these three ports are translated into AADL. Ports b and d and e are internal and thus not represented in the interface of the process. Yet, there are internal connections between e and c, and between b and d. Since no thread manages d, it is integrated into the threads of component B. Facet c is managed by an independent thread; therefore it is represented inside the AADL process.

The two threads that manage facet a of component B also manage facet d of component C, but not facet c. Hence, the two threads have three interfaces: a, e and f.

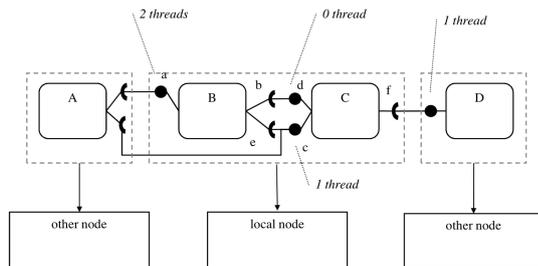
The thread that manages facet c of component C has two interfaces: c and f. It does not have any interface for d, since it does not manage this facet.

The AADL process encapsulates the hardware source code of components B and C, as well as the envelope code that connects the CIF to the AADL software runtime API. The threads call this source code.

CCM configuration:



AADL architecture:



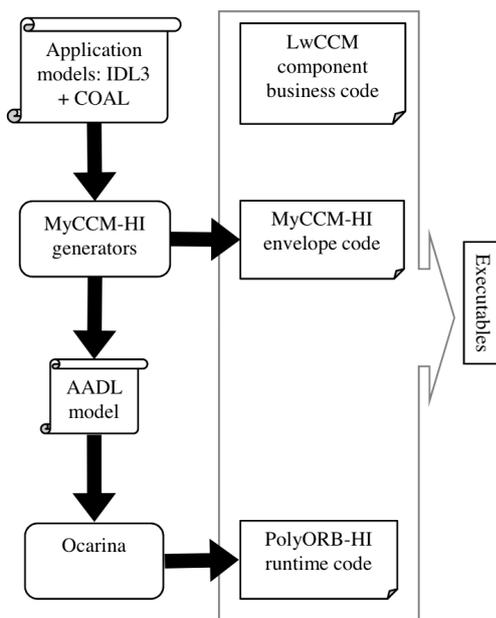
6 Implementation in MyCCM-HI and Ocarina

We described the mapping between a description made in IDL3 and deployment information, and AADL. This mapping has been implemented in a tool chain that allows the generation of C code from IDL3 and COAL files.

In this section, we explain how we implemented the CCM/AADL transformation rules. This mainly consists of the description of our tool chain.

6.1 Description of the Framework

The tool chain is the combination of MyCCM-HI and Ocarina.



Ocarina is an AADL compiler developed by Télécom ParisTech. It can generate C or Ada code from AADL descriptions. The generated code is a configuration layer for PolyORB High Integrity. PolyORB-HI is an AADL runtime; it provides thread management and communication primitives, according to the AADL semantics. PolyORB-HI is statically configured at code generation time. As a main consequence, no dynamic memory allocation is performed.

MyCCM High Integrity is a component framework developed by THALES. Its purpose is to generate AADL descriptions that can be processed by Ocarina, as well as the necessary source code for CCM component envelopes. The envelope code interfaces the component implementation code, written by the framework user, and the code generated by Ocarina for the AADL threads.

From the application model (i.e. IDL3 and COAL files), MyCCM-HI generates both envelope code in C and AADL architecture. This is the application of the mapping we described in the previous section.

The AADL architecture is then processed by Ocarina to generate C code that is meant to be compiled on the top of PolyORB-HI.

Finally, deployment code generated by Ocarina, envelope code generated by MyCCM-HI and component implementation provided by the user are compiled together (typically with gcc) to produce an executable binary.

6.2 Performances

In order to validate our approach, we made some benchmarks on the generated code. These benchmarks were based on a simple architecture: a typical message passing application, with a component on a node, sending a message to another component on another node. Each time it receives a message, the second component sends back a message to the first component. Thus we can calculate round-trip communication times.

For this architecture, the size of the PolyORB-HI runtime is approximately 20 kB; the size of a component envelope is about 4 kB. Such a small footprint validates our approach as a suitable solution for embedded applications, with little memory resource.

We made tests with two embedded boards, equipped with Freescale MPC5200 processors at 384 MHz, running the ElinOS real-time Linux kernel. We executed the application on the two boards while running some random load programs at the same time (successive memory allocation, file accesses, heavy computation, etc.). The benchmarks showed a latency of 768 μ s for the roundtrip, with a jitter of 24 μ s, that is, a jitter of 3.12%. It thus shows that the PolyORB-HI runtime and the component envelope generated by MyCCM-HI provide stable execution time, suitable for real-time applications.

7 Conclusion

In this paper, we explained a mapping between CCM and AADL architectures. CCM architectures are described with

IDL3 and deployment information. The mapping allows for the translation of a CCM application into an AADL architecture. Hence, the user code implementing the CCM components benefits from an API close to the Lightweight CCM standard, while it actually runs within an AADL application, with AADL semantics.

Thanks to this approach, we benefit the advantages of AADL: analysis capabilities and suitable performances of the generated code regarding real-time and embedded aspects. We also benefit the application design approach of Lightweight CCM, centred on business code.

We implemented this approach in a tool chain. This tool chain is made of two main tools, which are both free software. MyCCM-HI is a CCM framework that generates AADL architecture from CCM descriptions, and C envelope code. It can be downloaded from <http://myccm-hi.sf.net>. Ocarina generates C or Ada code for the PolyORB-HI runtime from AADL descriptions. It can be downloaded from <http://aadl.telecom-paristech.fr>.

8 Acknowledgement

The authors would like to thank Jérôme Hugues, Étienne Borde, Grégory Haïk and Bruno Hergott, who participated in the work we presented. They either developed parts of the theories we presented, contributed in the design of the tools, or implemented them.

References

- [1] SAE. Architecture Analysis & Design Language (AADL) v2.0 (proposed draft). Technical report AS5506A, SAE, 2009.
- [2] OMG Deployment & Configuration (D&C)
- [3] OMG CORBA Component Model specifications (version 4.0)
- [4] B. Zalila, L. Pautet, J. Hugues. Towards automatic middleware generation, 11th IEEE International Conference on Object-oriented Real-time distributed Computing (ISORC'08), 2008
- [5] P. Dissaux, HOOD Patterns, Data Systems in Aerospace (DASIA'01), 2001
- [6] SAE. AADL v2 Data Modeling Annex, draft v0.8, 2008
- [7] W. Emmerich, N. Kaveh, Component technologies: Java beans, COM, CORBA, RMI, EJB and the CORBA component model. 24th International Conference on Software Engineering (ICSE'02), 2002.
- [8] É. Borde, G. Haïk, L. Pautet. Scheduling Mode-based Reconfiguration in Critical Embedded Systems, Design, Automation & Test in Europe, (DATE'09), 2009
- [9] SPICES Project, D3.3.1: Mapping between Lightweight CCM and AADL, 2008

Ada User Guide on MaRTE OS

Mario Aldea Rivas, Michael González-Harbour

Universidad de Cantabria, 39005-Santander, Spain; email: {mgh, aldeam}@unican.es; http://martel.unican.es

Abstract

MaRTE OS (Minimal Real-Time Operating System for Embedded Applications) is a free software platform that allows to execute concurrent real-time Ada, C and C++ applications on a bare PC and, with some limitations, on a Linux box. It is mostly written in Ada with some C and assembler parts. Ada applications running on top of MaRTE OS can use the full Ada language functionality including most of the new services defined in the RM real-time annex. The GNAT run-time has been adapted to run on the POSIX interface provided by MaRTE OS. This guide provides information about MaRTE OS from the user's perspective, including the supported functionality and the installation and usage procedures on the different supported platforms.

1 Introduction

MaRTE OS is a real-time kernel for embedded applications that follows the Minimal Real-Time System Profile (PSE51) defined in the POSIX.13 standard [1] and is usable from Ada applications through an adaptation of the GNAT run-time system. Given the support for real-time functionality that MaRTE OS provides it can be used to develop advanced real-time applications according to the real-time annex defined in Ada 2005 [2][3]. The services provided by the system have a time-bounded response, so hard real-time requirements can be supported. MaRTE OS is distributed under a modified-GPL free-software license.

MaRTE OS was initially designed to support embedded applications running on a bare processor. Currently the supported architecture is a bare PC using an 80386 processor or higher. A basic hardware abstraction layer (HAL) is defined to facilitate porting to other architectures. In effect, this layer can be implemented using the services of another operating system that acts as a virtual processor. An implementation of MaRTE OS is available on the Linux operating system, which is useful for testing, development, and teaching purposes.

Most of the internal code of MaRTE OS is written in Ada with some C and assembler parts. Nonetheless, APIs for different programming languages are provided, allowing for the development of concurrent real-time applications written in Ada, C and C++. It is even possible to mix different languages in the same application, for instance with coexisting (and cooperating) C threads and Ada tasks running under a coherent real-time scheduling policy. In addition, Java (RTSJ) applications can be executed on MaRTE OS when it is configured as a POSIX threads (pthreads) library for Linux.

The development environment is based on the GNU compilers GNAT, gcc, and gcj, as well as on their associated utilities such as the gdb debugger.

When developing embedded applications, a cross development environment is used with the development tools hosted on a Linux system. The executable images can be uploaded to the target via an ethernet link, and cross debugging is possible through a serial line. It is also possible to write the executable image to a bootable device such as a flash memory, for isolated execution in the target.

MaRTE OS has been used to develop industrial embedded systems and is also an excellent tool for educational activities related to real-time embedded systems programming.

This paper will give a high-level user perspective of MaRTE OS for Ada programmers, and is intended for people interested in evaluating the system, or using it for developing embedded applications or for teaching real-time programming.

The paper is organized as follows. Section 2 gives some details on the architecture of MaRTE OS that are useful to understand its principles of operation. Section 3 describes the properties and installation procedures in a Linux platform, while Section 4 does the same for a bare PC platform. Section 5 describes the supported functionality, and the usage procedures appear in Section 6. Finally, Section 7 discusses future work.

2 Architecture

The central part of MaRTE OS is its kernel, which implements the support for the concurrency services and the functionality described in Section 5.

The kernel has a low-level abstract interface for accessing the hardware. This hardware abstraction layer (HAL) encapsulates operations for interrupt management, clock and timer management, and thread context switches. Its objective is to facilitate migration from one platform to another, being the implementation of this hardware abstraction layer the only part that needs to be modified in that process.

The kernel provides a POSIX interface [4] through a collection of Ada functions with the same profiles as the C language POSIX functions. This approach implies two important advantages:

- C applications can use the kernel directly, just a set of C-header files are needed.

- The GNAT run-time system for Linux is layered on top of the C POSIX interface. So, it was easy to adapt it to run on our kernel.

The operating system kernel is just one of the pieces that is used in the MaRTE OS environment to execute Ada applications. Other important components are the GNAT run-time system, the C standard library and the device drivers. MaRTE OS also provides a POSIX-Ada interface (POSIX.5b) to facilitate synchronization between Ada tasks and C threads.

Figure 1 shows the relationship among all these components for an Ada application running on MaRTE OS in a PC computer.

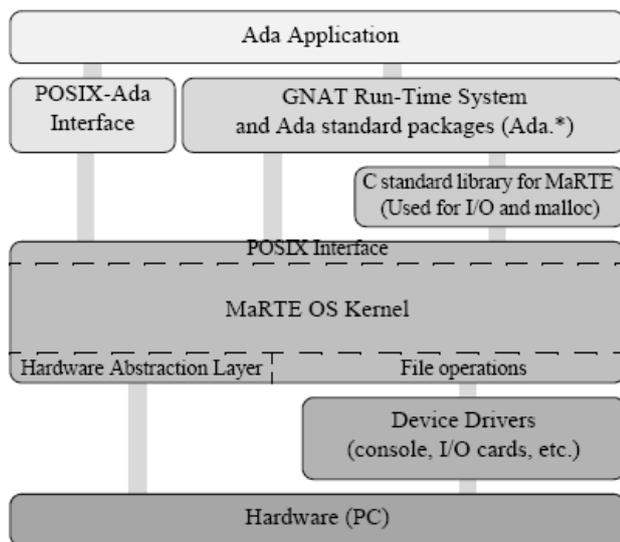


Figure 1 MaRTE OS Architecture

Although originally intended for embedded systems based on the PC architecture, MaRTE OS has been also adapted to behave as a POSIX-threads library for Linux. So currently MaRTE OS can be used on the following platforms:

- MaRTE on a bare PC (architecture “x86”). MaRTE OS applications are stand-alone programs that can be launched on a bare PC.
- MaRTE on Linux (Architectures “linux” and “linux_lib”). MaRTE OS applications are executed as any other standard Linux user process.

3 MaRTE on Linux

When running on Linux, MaRTE OS is used to provide concurrency at the library level to Ada and C applications. In the case of Ada, MaRTE OS is used as the POSIX-threads library that supports the Ada tasking services implemented by the run-time system of the GNAT compiler.

MaRTE OS provides two different alternatives to execute on Linux, which are treated as different architectures in the MaRTE installation process but that, in fact, are very similar. They are called “linux” and “linux_lib” architectures and they differ in that the “linux_lib”

architecture uses the Linux libraries and file system, while the “linux” architecture uses the standard C library provided by MaRTE OS, and thus its own internal pseudo file system.

The main differences between the architecture for bare PCs and the “MaRTE on Linux” architectures are in the hardware abstraction layer:

- In “MaRTE on Linux” a Linux timer is used instead of the hardware timer of the “x86” architecture.
- In “MaRTE on Linux” Linux signals play the role of the hardware interrupts in the “x86” architecture.
- The context switch routine is the same for the “linux”, “linux_lib” and “x86” architectures.

The process of building an application for the “linux_lib” architecture is shown in Figure 2. In first place, the user’s code files are compiled. Next, the obtained object files are linked with MaRTE and other static or dynamic Linux libraries in order to build a standard Linux program.

“MaRTE on Linux” applications are executed as any other standard Linux program and they can be debugged using gdb or any other standard debugger.

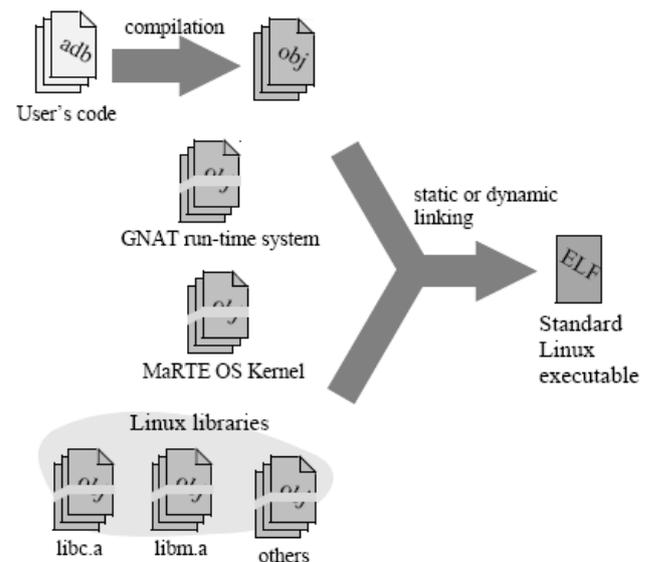


Figure 2 Building an application for the “linux_lib” architecture

Although MaRTE on Linux is a very convenient solution in some cases, it has its limitations:

- Hard real-time behaviour cannot be achieved since MaRTE applications are standard user processes that share CPU time with all the other processes in the system according to the Linux scheduling policies. As any other process, they are affected by memory swapping, Linux kernel activities, etc.
- Measurement of execution time of tasks is inaccurate because context switches between the MaRTE application and other Linux processes are not taken into account. Accuracy is improved if there are no other user processes

running in the same box, but there can still be system processes active.

- A common limitation of library-level concurrence is the global blocking on I/O operations. Since only one Linux thread is used to implement all the user's tasks, a task waiting for an I/O operation will imply the blocking of the only Linux thread and, consequently, the whole application will block.

Despite those limitations, MaRTE OS on Linux represents a good choice to teach real-time Ada (or POSIX) programming courses, because of the following advantages:

- Correct behaviour of priorities and scheduling policies.
- New Ada 2005 real-time services: execution time clocks and timers, task group execution time budgets, timing events, dynamic ceiling priorities for protected objects and additional scheduling policies: round robin, EDF, and priority-specific dispatching.
- All in an inexpensive platform: free software in a standard Linux box.

Apart from teaching activities, the “linux” and “linux_lib” architectures can also be used as a fast mechanism to perform preliminary functional testing of applications before their definitive testing in the embedded computer.

3.1 Architecture “linux”

This architecture is intended to be as close as possible to the “x86” architecture. With that purpose, the C standard library used is the one adapted for MaRTE (instead of the standard library provided with the Linux operating system). This implies the MaRTE pseudo-file system is used and dynamic memory management is performed by the `malloc()` and `free()` functions provided by MaRTE, which implement the TLSF algorithm [5] that has efficient time-bounded allocation and deallocation.

This architecture only provides simple drivers for the console and the keyboard. They just put and get characters to and from the Linux “`stdout`” and “`stdin`” file descriptors (as opposite to managing the hardware devices directly like these drivers do in the “x86” architecture).

The structure of an Ada application running on the “linux” architecture is shown in Figure 3. As it can be seen it is very similar to the one shown in Figure 1, only changing the drivers and the hardware abstraction layer.

Due to its similarity with the “x86” architecture, this architecture is the most appropriate to perform preliminary testing of applications before their final testing in the embedded computer.

3.2 Architecture “linux_lib”

In this architecture MaRTE OS only behaves as a POSIX-threads library that supports Ada tasking. Other POSIX services not related to threading are delegated to the Linux C standard library. In particular, the use of this library implies that the dynamic memory management is carried

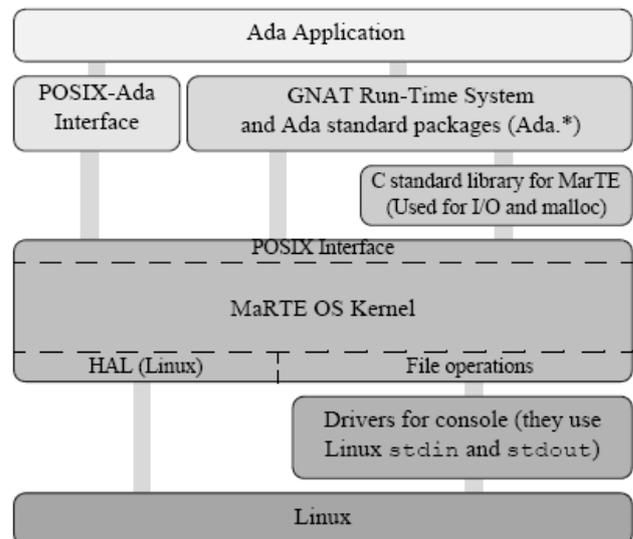


Figure 3 The “linux” architecture

out by Linux. The second implication, probably more interesting for the users, is that the Linux file system and other system features can be accessed from MaRTE applications using the standard POSIX, Linux or Ada APIs like in any other Linux program.

The structure of an Ada application running on the “linux_lib” architecture is shown in Figure 4. The main difference with Figure 3 is the use of the Linux C standard library in place of the MaRTE C library and drivers.

This architecture is the most appropriate for those who want to write Ada program that use the Linux services and, at the same time, take advantage of the full Ada scheduling and the new Ada 2005 real-time services provided by MaRTE OS.

3.3 Installation

This subsection will provide an overview of the MaRTE OS installation process for the “MaRTE on Linux” architectures. For detailed information read the `INSTALL` document included in the MaRTE tarball.

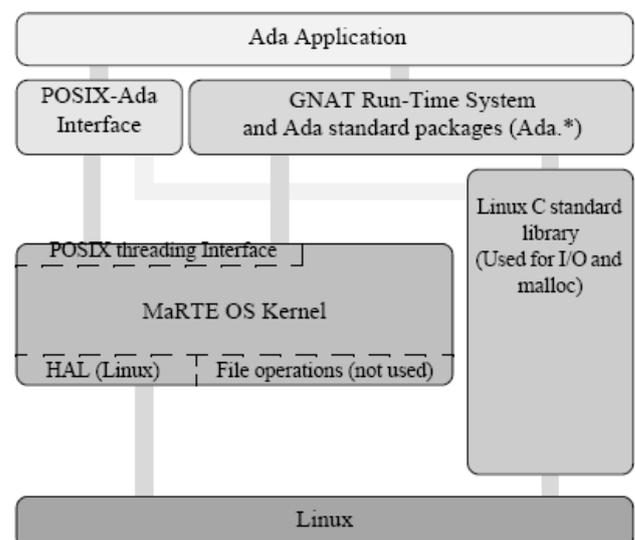


Figure 4 The “linux_lib” architecture

Installing “MaRTE on Linux” is straightforward. You only need a Linux computer with a working installation of the GNAT-GPL-2009 compiler. Probably the easiest way of installing MaRTE is downloading the binary tarball from the MaRTE OS web page. Just unpack the tarball and run the mininstall script.

Just after running the mininstall script you will have a working MaRTE installation set to use the “linux_lib” architecture. At this point it is already possible to make and execute our first “MaRTE OS on Linux” program:

```
terminal
$ cd examples/
$ mgnatmake hello_world.adb
$ ./hello_world

Hello, I'm an Ada program running on MaRTE OS.
```

If you prefer to use the “linux” architecture you only need to run the msetcurrentarch command (more on MaRTE utilities in Section 6):

```
terminal
$ msetcurrentarch -march linux
```

The applications compiled after executing that command will use the chosen architecture.

4 MaRTE OS on a bare PC

When using this architecture, MaRTE OS applications are stand-alone programs that can be launched on a bare PC. This architecture is called “x86” in the MaRTE installation process.

For “PC” we mean any “PC compatible” computer: netbook, laptop, desktop, or embedded computer (e.g., a PC104 single board computer) with a processor that is binary compatible with 80386, 80486, Pentium, P-II, P-III or P-IV processors. The computer should also have the standard PC devices: Programmable Interrupt Controller (PIC), Programmable Interval Timer (PIT), Real-Time Clock (RTC), etc. In fact MaRTE OS requirements are very standard and we have not found any PC-like computer where MaRTE can not be run.

Going more into the details, a MaRTE program for this architecture is a multiboot-compliant x86 ELF executable that can be booted using any multiboot loader (e.g., GRUB, the loader used by Linux). In this program the user’s code is statically linked with all the libraries required to execute in a bare PC: MaRTE OS kernel, drivers, C standard library, etc. In the linking phase the code that performs the initialization of the system is also included. This piece of code will be the first to be executed once the application is launched in the target. When the initialization finishes, the main user’s procedure is executed. The process of building an application is shown in Figure 5. The structure of an Ada application running on the “x86” architecture was shown in Figure 1.

4.1 Cross-development environment

An embedded systems development cycle is usually performed in a cross-development environment. The cross-

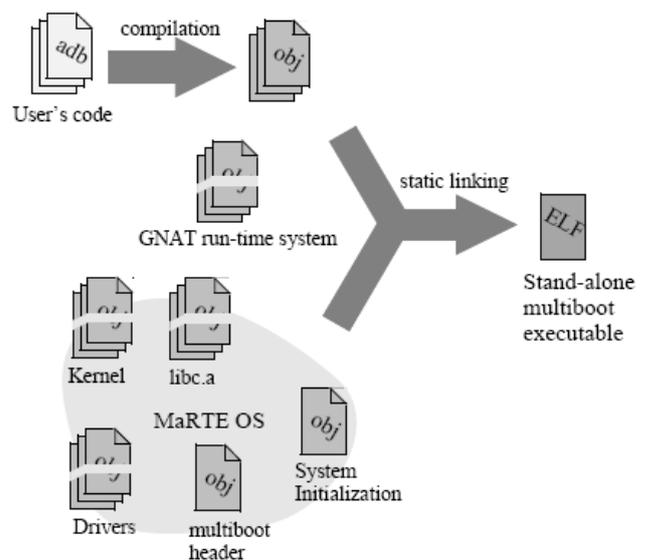


Figure 5 Building an application for the “x86” architecture

development environment for MaRTE OS (architecture “x86”) is shown in Figure 6. It is formed by a PC running Linux as “Host” and a bare x86 PC as “Target”, with both systems connected by an Ethernet LAN for application downloading, and a serial line for remote debugging using the GNU debugger gdb.

The application is built in the host computer using the GNAT compiler and the scripts provided with MaRTE (described in Section 6). Once the application has been built it is downloaded to the target through the ethernet and executed there. Details about this process are provided in Section 4.4.

4.2 Target processor and timing services

The implementation of clocks and timing services in MaRTE OS depends on the processor available in the target computer.

If the processor is an 80386 or 80486, the Programmable Interval Timer (PIT) is used both for the clocks (i.e., for

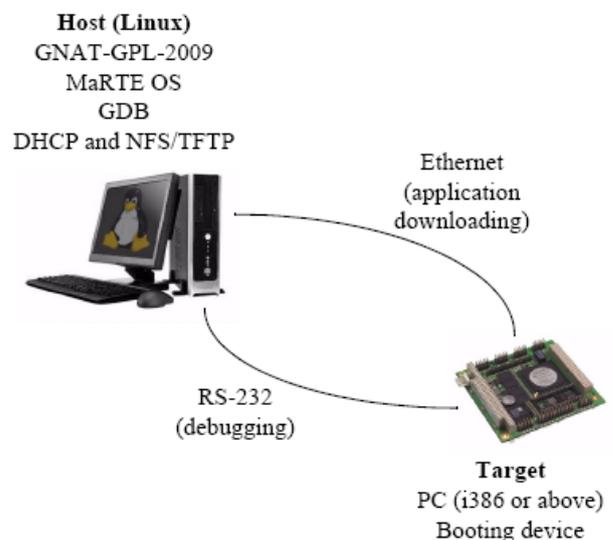


Figure 6 MaRTE OS cross development environment

measuring absolute or system time) and for the timing services (i.e., those requiring the generation of a timed event at the requested time). The PIT is a standard device in the PC architecture that has three 16-bit counters driven through a hardware clock signal of 838.1 ns period. The main problem with the PIT is that its registers are accessed through the old I/O bus in the PC architecture, which makes accessing any of these registers a very slow operation.

If a Pentium processor is available, the measurement of absolute time can be implemented using the time-stamp counter (TSC). This counter (as implemented in the Pentium and P6 family processors) is a 64-bit counter that is set to zero following the hardware reset of the processor. Following reset, the counter is incremented every processor clock cycle. Reading the value of this counter requires only a single machine instruction and, because this counter is internal to the processor and the I/O bus is not used, the operation is very fast. In this implementation, timer interrupts are still generated with the PIT's Counter 0.

For P6 processors (Pentium II or higher) the overhead of the timing services can be greatly diminished by using the timer included in the Advanced Programmable Interrupt Controller (Local APIC). The local APIC is included in all P6 family processors. Although its main function is the dispatching of interrupts, it also contains a 32-bit programmable timer for use by the local processor whose time base is derived from the processor's bus clock.

Accessing the TSC or the Local APIC is much faster than using the PIT. Table 1 shows the comparison of the overheads of the different time services implementations on a target computer with a Pentium III at 500 MHz.

In addition to being more efficient, the resolution of the time services improves when using more modern devices like the TSC and Local APIC, as can be seen in Table 2.

Table 1 Comparison of the overheads of the time services implementations

Operation	PIT	TSC+PIT	TSC+APIC
Get time	3100 ns	105 ns	105 ns
Program timer	12900 ns	3000 ns	324 ns

Table 2 Comparison of the resolution of the time services implementations

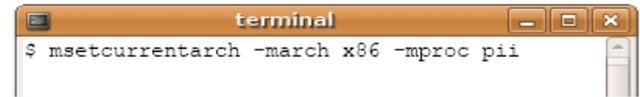
Service	PIT	TSC+PIT	TSC+APIC
Clock	838.1 ns	2 ns (1 processor cycle)	2 ns (1 processor cycle)
Timer	838.1 ns	838.1 ns	10 ns (1 processor's bus clock cycle)

4.3 Installation of MaRTE OS in the host computer

The basic requirements to install “MaRTE on a bare PC” in the host computer are the same than for “MaRTE on Linux”: a Linux computer with a working installation of the GNAT-GPL-2009 compiler. Just unpack the MaRTE

OS binary tarball and run the mininstall script. At this point you will have a working MaRTE installation set to use the “linux_lib” architecture.

Using the msetcurrentarch script it is possible to set “x86” as the default architecture (more on MaRTE OS scripts in Section 6). For example, if you want to use the “x86” architecture for a target with a Pentium II processor run the following command:



```
terminal
$ msetcurrentarch -march x86 -mproc pii
```

You can now build an application for the “x86” architecture using the mgntatmake script. A executable file called hello_world will be created:



```
terminal
$ cd examples/
$ mgntatmake hello_world.adb
```

But this just created executable is not intended to be run on Linux. You will obtain an error if you try to execute it that way:



```
terminal
$ ./hello_world
Segmentation fault
```

The program is trying to use the whole memory map of the computer and to access the hardware directly, both operations are not allowed in a protected operating system as Linux and causes the segmentation fault. We have to run our executable in a bare PC different from the host. This computer is called the target computer in our cross-development environment.

4.4 Setting up a cross-development environment

As we saw in Figure 6, our development environment has a host and a target computer. The application is built in the host and executed in the target; consequently, we need a mechanism to transfer it from one computer to the other.

There are several alternatives to set up a cross-development environment. The most convenient alternative is to use the network to communicate host and target, and this involves configuring protocols like DHCP, NFS and/or TFTP in the host and it also requires a booting device in the target (floppy, hard disk, flash RAM, PXE, etc.).

Lets briefly describe one of the simplest mechanisms to download the application from the host computer to the target. It is based on a target with floppy disk and uses the Etherboot network loader [6].

First of all, you need an Etherboot floppy bootable ROM image compatible with the Ethernet card of your target. It can be easily generated at <http://rom-o-matic.net/>. The image should be set to use the DHCP and NFS protocols.

The host should be configured as a DHCP server; its configuration file (usually named as dhcpd.conf) should be modified to add an entry with the ethernet MAC address of the target computer together with its assigned IP address and the location (directory and name) of the program to be

executed. Similarly, the host should be configured as an NFS server, configured to export the program location. The executable file produced with the `mgnatmake` script should be copied to the exported directory.

The process of booting the target computer is shown in Figure 7. When the target is started it boots from the floppy and, at this point, the Etherboot network loader takes over the computer. The first step of Etherboot is to broadcast a DHCP request asking for its IP address and the name and directory of the program to be downloaded. The host has been configured to reply to that request providing the desired information. Then, Etherboot downloads the MaRTE program from the given location using the NFS protocol and places it at the appropriate position in memory. Next, Etherboot passes the control to the program, which performs the initialization of the target computer and, finally, invokes the user's main procedure.

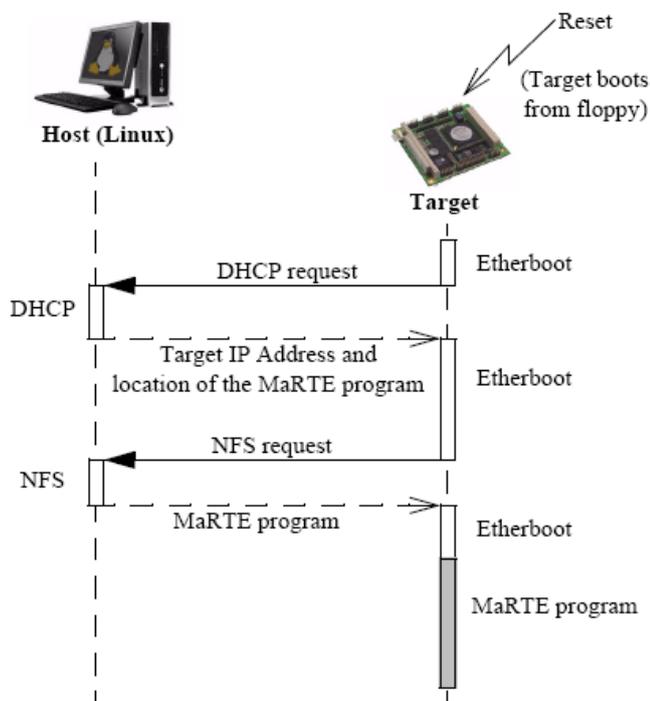


Figure 7 Target boot process

For a more extensive information on the booting process including instructions about how to configure the protocols in the host, read the “MaRTE OS Boot process (x86 architecture)” document in the MaRTE OS web page. This document also details how to set up a cross-development environment for a target with hard disk, flash RAM or PXE.

5 Supported functionality

MaRTE OS is an implementation of the POSIX.13 minimal real-time system profile and as such it provides the services defined in the standard, which can be grouped as:

- Concurrency services supporting the management of threads.
- Scheduling with real-time policies based on preemptive fixed priorities, and supporting variants such as FIFO

within priorities, round robin within priorities, or the sporadic server.

- Synchronization through counting semaphores, mutexes, and condition variables. Mutexes have support for real-time mutual exclusion through the priority inheritance or priority ceiling protocols.
- Signals, as an asynchronous notification mechanism.
- Time management through clocks and timers. A monotonic clock that cannot have backward jumps is provided for real-time applications. Timers can be created to measure the passage of an absolute or relative time and will generate a signal to notify the application about their expiration.
- Execution-time clocks and timers are used to measure execution time, which is needed for real-time analysis, and to monitor and bound the usage of execution time, which is crucial to ensure that the results of the schedulability analysis hold during execution.
- Dynamic memory management, MaRTE OS uses the TLSF [5] algorithm developed at the Technical University of Valencia, which is a fast time-bounded dynamic memory allocator with low fragmentation.
- Device I/O through a simplified device name space and the standard open/close/read/write/ioctl operations.

In addition, the following service is not defined by the POSIX standard (due to the difficulty for defining a fully portable API), but is provided as a necessary service for embedded applications:

- Interrupt management with the ability to install interrupt handlers and manage interrupt masking. In MaRTE OS there is separate accounting for the execution time of interrupt handlers.

Most of these services are accessed through the equivalent services defined in the Ada standard and thus the application developer will use them transparently, without any need for a specific knowledge. For instance, Ada tasks are mapped to OS threads and are managed and scheduled through the Ada language constructs. Mutexes and condition variables are used to implement Ada's protected objects, and therefore synchronization is also achieved through standard Ada constructs. The same happens with time management, interrupt management, and dynamic memory management, for which the Ada services are transparently mapped to the corresponding MaRTE OS services by the implementation. Execution time clocks and timers are accessed through the API defined in Ada's Real-Time Annex.

In the case of mixed-language applications, if interoperability is required between Ada and non-Ada threads it is necessary to directly use the OS synchronization primitives (mutexes and condition variables) from the application level. Foreign threads should not use protected objects because the Ada runtime system is unaware of them and will not be able to provide the correct support. The usage of mutexes and condition

variables from the Ada application is done following the services defined in the standard POSIX-Ada bindings [7].

In addition to the POSIX services, MaRTE OS also provides extensions that are useful to develop advanced real-time applications. Some of these extensions are used to implement the new real-time services defined in Ada 2005, such as:

- Timed handlers, as a lightweight mechanism to define small handlers that are executed in interrupt context at the expiration of a timer. These handlers can also be used in conjunction with execution time clocks.
- Earliest-Deadline-First (EDF) scheduling. This is a preemptive dynamic-priority thread dispatching policy that can be used to maximize the resource utilization in real-time applications. It requires an implementation of Baker's protocol [8] for avoiding unbounded blocking effects in accessing protected objects.
- Priority-specific dispatching, allowing two-level scheduling policies, in which several policies can coexist. This is specially useful in applications mixing different kinds of timing requirements such as: critical hard real-time tasks that need to be scheduled under fixed priorities for predictability purposes; soft real-time tasks that require high levels of utilization and therefore need an EDF scheduler running at the intermediate priority bands; and non-real time tasks running at a low priority level under a round-robin scheduler that provides fairness among them.
- The ability to create thread sets or groups, to be used for instance to create clocks that measure the execution time of a group of threads. These clocks can in turn be used to create timers and timed handlers to implement advanced scheduling policies or to detect and bound the effects of timing violations by a group of threads.

With these services MaRTE OS provides full support for the Ada 2005 real-time services with the exception of non-preemptive scheduling.

Other extensions are provided by MaRTE OS which are not defined in the Ada standard. The most important one is application-defined scheduling [9], which is a group of services intended to allow an application to install its own scheduler for the OS threads (and therefore for Ada tasks). This is particularly interesting to implement advanced scheduling policies being defined by the real-time research community.

6 Usage

In the `utils/` directory, MaRTE OS provides a set of scripts that allows building applications and configuring and compiling MaRTE and its related libraries. When using MaRTE it is convenient to have this directory in the `PATH` environment variable in order to have direct access to these basic commands.

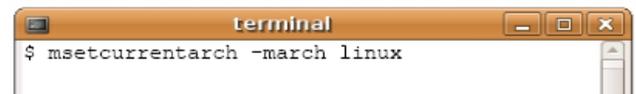
6.1 Setting the current architecture

As we seen before, using the same installation of MaRTE we can create applications for the three architectures:

“linux”, “linux_lib” and “x86”. The active architecture (i.e., the one the applications are going to be generated for) is chosen using the `msetcurrentarch` script.

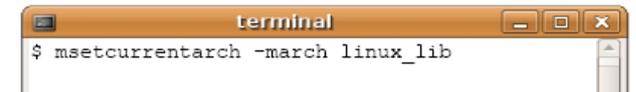
When used without parameters, `msetcurrentarch` returns the current and the available architectures.

The main use of the `msetcurrentarch` script is to change the current architecture. In order to configure “linux” as the default architecture execute:



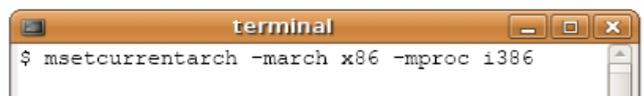
```
terminal
$ msetcurrentarch -march linux
```

To choose “linux_lib”:



```
terminal
$ msetcurrentarch -march linux_lib
```

And to choose “x86”:



```
terminal
$ msetcurrentarch -march x86 -mproc i386
```

For the “x86” architecture, the `-mproc` flag allows specifying the processor to be used:

- `i386`: any Intel x86-compatible processor. The PIT is used for the timers and for the clock.
- `pi`: Pentium I or above. The PIT is used for the timers and the TSC is used as the clock.
- `pii`: Pentium II or above. The LocalAPIC timer is used for timers and the TSC is used as the clock.

6.2 Making applications

The main script to build an Ada application is `mgnatmake`. This script is the MaRTE equivalent to the standard `gnatmake` command for the GNAT compiler. The script can be invoked with the same arguments that are used with `gnatmake`. Examples of valid `mgnatmake` invocations are:



```
terminal
$ mgnatmake -gnato my_program.adb
```



```
terminal
$ mgnatmake -g my_program -larges obj.o
```

In these examples the `-gnato` option enables overflow checking. The `-g` option specifies that debugging information should be added to the executable file, while `-larges obj.o` tells the linker to also link the specified object file.

Internally the `mgnatmake` script invokes `gnatmake` with the appropriate arguments and binder and linker switches depending on the current architecture. In particular, `mgnatmake` links the user's application code with the appropriate MaRTE OS and/or Linux libraries as shown in Figure 2 and Figure 5.

In case you need to perform compiling, binding and linking phases independently, MaRTE OS also provides the `mgnatbind` and `mgnatlink` scripts.

The other basic script to make applications is `mgcc`. It is the MaRTE replacement for `gcc` and, consequently, it can be used to compile Ada files and, of course, to compile and build C applications:

```
terminal
$ mgcc -g -O2 extra_obj.o my_program.c -lm
```

6.3 Recompiling MaRTE OS

When MaRTE OS is installed from the binary tarball the kernel and libraries are compiled using full optimization options. For most users this default configuration is enough and they do not need to recompile MaRTE OS again. However, there are some reasons why an advanced user could be interested on recompiling MaRTE OS:

- In order to change the maximum number of resources allowed in a MaRTE application and other configuration parameters.
- In case you are making some modifications to MaRTE OS or adding some new functionality.
- If you want to compile the kernel with some compiler switches different than the default ones.
- If you want to enable some debug checks and messages in the kernel.
- If you are installing a new device driver in the system.
- If you want to use the “`tasks_inspector`” tool which allows obtaining a graphical trace of the task execution.

In the “MaRTE OS User’s Guide” included in the MaRTE tarball you will find information about configuration parameters, device drivers, debug checks, etc.

The script provided by MaRTE to compile the kernel, libraries and drivers is `mkmarte`. This script accepts `gnatmake` and/or `gcc` switches.

For example, to compile MaRTE for the current architecture with debug information and assertions enabled you can execute:

```
terminal
$ mkmarte -g -gnata
```

MaRTE OS also provides the `mkrtsmarteuc` script to recompile the GNAT run-time for the current architecture. For example to recompile the run-time system with debug information you can execute:

```
terminal
$ mkrtsmarteuc -g
```

7 Future work

The current implementation of MaRTE OS is designed to run in single processors. Although MaRTE applications can run in multicore platforms, only one of its cores will be used. Since multicore platforms are becoming so usual, it is important to redesign MaRTE OS to take full advantage of such platforms. Therefore one of the main objectives for

future development of MaRTE OS has been set towards this end.

Migration to other execution platforms is also in the agenda for future work. In particular, the ARM family of processors is quite popular in the embedded systems world and is a good candidate for this migration effort.

Networking is available in MaRTE OS through real-time protocols such as RT-EP or CAN-RT-TOP, but the implementations of these protocols use special-purpose APIs. It would be useful to implement POSIX sockets as the API to access these networking services.

Finally, a simple extension is the addition of Ada’s non-preemptive task dispatching policy, which would allow us to provide complete support for the Ada 2005 Real-Time Annex.

Acknowledgements

The authors would like to thank AdaCore for their support and guidance, and for the help in the adaptation of the run-time system. Special thanks also to all the many contributors to MaRTE OS.

References

- [1] IEEE Std. 1003.13-2003. Information Technology - Standardized Application Environment Profile-POSIX Realtime and Embedded Application Support (AEP). The Institute of Electrical and Electronics Engineers.
- [2] Mario Aldea Rivas, José F. Ruiz. “Implementation of new Ada 2005 real-time services in MaRTE OS and GNAT”. Lecture Notes on Computer Science (Vol: 4498), 29-40. June, 2007
- [3] M. Aldea, M. González Harbour, José F. Ruiz. “Implementation of the Ada 2005 Task Dispatching Model in MaRTE OS and GNAT”. 14th International Conference on Reliable Software Technologies - Ada-Europe 2009. Brest, France. June, 2009.
- [4] IEEE Std. 1003.1:2004 Edition, Information Technology —Portable Operating System Interface (POSIX). The Institute of Electrical and Electronics Engineers.
- [5] TLSF (Two-Level Segregate Fit) allocator. <http://rtportal.upv.es/rtmalloc/>
- [6] Etherboot network bootloader. <http://etherboot.org/wiki/>
- [7] POSIX.5b (1996). IEEE Std 1003.5b-1996, Information Technology — POSIX Ada Language Interfaces — Part 1: Binding for System Application Program Interface (API) — Amendment 1: Realtime Extensions. The Institute of Electrical and Engineering Electronics.
- [8] Baker T.P., “Stack-Based Scheduling of Realtime Processes”, *Journal of Real-Time Systems*, Volume 3, Issue 1 (March 1991), pp. 67–99.
- [9] Mario Aldea Rivas and M. González Harbour. “A New Generalized Approach to Application-Defined Scheduling”. *Proceedings of 16th Euromicro Conference on Real-Time Systems (WiP)*, Catania, Sicily (Italy), July 2004.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
 c/o K.U. Leuven
 Dept. of Computer Science
 Celestijnenlaan 200-A
 B-3001 Leuven (Heverlee)
 Belgium
 Email: Dirk.Craeynest@cs.kuleuven.be
 URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
 Email: Info@Ada-DK.org
 URL: Ada-DK.org

Ada-Deutschland

Dr. Peter Dencker
 Steinäckerstr. 25
 D-76275 Ettlingen-Spessart
 Germany
 Email: dencker@web.de
 URL: ada-deutschland.de

Ada-France

Ada-France
 attn: J-P Rosen
 115, avenue du Maine
 75014 Paris
 France
 URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
 Ada-Spain
 P.O.Box 50.403
 28080-Madrid
 Spain
 Phone: +34-942-201-394
 Fax: +34-942-201-402
 Email: gutierjj@unican.es
 URL: www.adaspain.org

Ada in Sweden

attn. Rei Strähle
 Saab Systems
 S:t Olofsgatan 9A
 SE-753 21 Uppsala
 Sweden
 Phone: +46 73 437 7124
 Fax: +46 85 808 7260
 Email: Rei.Strahle@saabgroup.com
 URL: www.ada-sweden.org

Ada Switzerland

attn. Ahlan Marriott
 White Elephant GmbH
 Postfach 327
 8450 Andelfingen
 Switzerland
 Phone: +41 52 624 2939
 e-mail: ada@white-elephant.ch
 URL: www.ada-switzerland.ch