ADA USER JOURNAL

Volume 31
Number 2
June 2010

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication else-where.

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

With this issue the Ada User Journal returns to the usual publication model, after a very successful 30th anniversary issue put forward by a prominent group of Guest Editors.

As usual, the June issue of the Journal is finalised shortly after the Ada-Europe conference, which took place this year in Valencia, Spain, in the week of June 14 to 18. The organizers must be congratulated for a very successful conference, with a very rich program, and, also important, a pleasant social program. As announced during the conference, next year the Ada-Europe conference will take place in Edinburgh, Scotland, combined with the Ada-Conference UK 2011, in an event under the name of *The Ada Connection*. It will be undoubtedly a very important event for Ada practitioners and enthusiasts and, I am certain, a great opportunity for the community to connect. You can find the preliminary call for papers in the Forthcoming Events section of this issue, together with the announcement and highlights of SIGAda 2010, which will take place this year in the Washington DC Area, USA, in the week of October 24 to 28.

This issue also includes two other announcements. The first relates to the annual Student Programming Contest being launched by Ada-Europe. This contest, with the title of *The Ada Way,* a yearly competition among teams of students, will be overseen by a steering committee composed of representatives of promoting institutions and evaluated by a panel composed of leading Ada experts. If you are an educator, please consider promoting the participation of a team from your school.

The second is a letter to the community concerning the current process of maintenance and revision of Ada, from WG9, the group responsible for the language. WG9, together with the ARG, the Rapporteur Group supervising the evolution of Ada, draws the attention of the community to the activities that are being carried out, and encourages all of us to actively participate in the current (and future) evolution of the language.

As for the technical contents of the issue, the first article, by authors from the Technical University of Madrid, Spain, provides the experience of developing device drivers with the Ravenscar profile, in the context of the Open Ravenscar real-time Kernel (ORK). The second article, from authors coming from SOGILIS, France, describes an agile-based development process to address requirements traceability.

The last paper of the issue is the first coming from the Industrial Track of the Ada-Europe 2010 conference. In it, Maciej Sobczak, from CERN, Switzerland, presents an approach to program callbacks in Ada bindings to C++ libraries.

To finalise, the Ada Gems section provides the gems with the Tokeneer Discovery Lessons, six lessons to explore the capabilities of the SPARK toolset. The News and Calendar sections complete the issue.

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

Quarterly News Digest

Marco Panunzio

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

Contents

Ada-related Events

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

Ada-Belgium Spring 2010 Event

From: Dirk Craeynest <dirk@asgard.cs.kuleuven.be>
Date: Tue, 11 May 2010 22:20:02 +0200
Subject: Ada-Belgium Spring 2010 Event, incl. Debian packaging workshop
Newsgroups: comp.lang.ada, fr.comp.lang.ada

A d a - B e l g i u m   S p r i n g   2 0 1 0   E v e n t

Saturday, June 5, 2010, 12:00-19:00

Leuven, Belgium

including at 14:00

2010 Ada-Belgium General Assembly

and at 15:00

Workshop on Creating Debian Packages of Ada Software

<http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/local.html>

Announcement

The next Ada-Belgium event will take place on Saturday, June 5, 2010 in Leuven.

For the third year in a row, Ada-Belgium decided to organize their "Spring Event", which starts at noon, runs until 7pm, and includes a barbecue, a key signing party, the 17th General Assembly of the organization, and a workshop on packaging Ada software for Debian hosted by Ludovic Brenta, principal maintainer of Ada in Debian.

Schedule

- 12:00 welcome and getting started (setting up computers and preparing food - please be there!)
- 13:00 barbecue
- 14:00 Ada-Belgium General Assembly
- 14:45 key signing party
- 15:00 workshop on creating Debian packages of Ada software
- 19:00 end

Participation

Everyone interested (members and non-members alike) is welcome at any or all parts of this event.

For practical reasons registration is required. If you would like to attend, please send an email before Tuesday, May 25, to Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be> with the subject "Ada-Belgium Spring 2010 Event", so you can get precise directions to the place of the meeting.

If you are a member but have not renewed your affiliation yet, please do so by paying the appropriate fee before the General Assembly (you have also received a printed request via normal mail). If you are interested to become a new member, please register by filling out the 2010 membership application form[1] and by paying the appropriate fee before the General Assembly.

After payment you will receive a receipt from our treasurer and you are considered a member of the organization for the year 2010 with all member benefits[2]. Early renewal ensures you receive the full Ada-Belgium membership benefits (including the Ada-Europe indirect membership benefits package).

As mentioned at earlier occasions, we have a limited stock of documentation sets and Ada related CD-ROMs that were distributed at previous events. Most important are back issues of the Ada User Journal[3]. These will be available on a first-come first-serve basis at the General Assembly for current and new members.

[1] http://www.cs.kuleuven.be/~dirk/ada-belgium/forms/member-form10.html

[2] http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html

[3] http://www.ada-europe.org/journal.html

Barbecue

The organization will provide food and beverage to all Ada-Belgium members. Non-members who want to participate at the barbecue are also welcome: they can choose to join the organization or pay the sum of 10 Euros per person to the Treasurer of the organization.

General Assembly

All Ada-Belgium members have a vote at the General Assembly, can add items to the agenda, and can be a candidate for a position on the Board[4]. See the separate official convocation[5] for all details.

[4] http://www.cs.kuleuven.be/~dirk/ada-belgium/board/

[5] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/10/100605-abga-conv.html

Key Signing Party

Wouldn't it be nice if a majority of people used GPG to sign their email every day so that you could send all non-signed email into the spam bin? To make that dream come true, please join and expand the global Web of Trust[6]!

What you should bring with you:

- an official ID card issued by your national government;
- your GPG key fingerprint (i.e. the output of gpg --fingerprint) on small paper slips; a dozen copies or so should be enough.

What you will go home with:

- signatures from all other participants;
- automatic inclusion in the global Web of Trust;
- the ability to digitally sign or encrypt anything you like.

[6] http://en.wikipedia.org/wiki/Web_of_Trust

Workshop: Packaging Ada Software for Debian

Debian[7], "The Universal Operating System", is simply the best platform for the enthusiast Ada developer. The features that distinguish Debian from the rest are:

- a binary distribution that avoids the need to recompile Florist, ASIS, GtkAda and all other Ada packages;
- a large number of packages intended for Ada developers;
- a clear and consistent policy[8] making all packages integrate seamlessly and interoperate;
- outstanding support for the Ada part of the GNU Compiler Collection (GCC) with unique innovations like libgnatvsn and libgnatprj not found anywhere else;
- backports of bug fixes from the bleeding edge of GCC development into the safe and stable compiler used for all Debian packages;
- support for more hardware architectures than any other Ada distribution: alpha, amd64, hppa, i386, ia64, kfreebsd-i386, powerpc, s3980 and sparc (with mips, mipsel and ppc64 added recently).
- a choice between "stable", "testing" and "unstable" versions of Debian to suit personal preferences;
- Debian is the mother of Ubuntu, Knoppix and dozens of other distributions which sometimes incorporate the Ada packages.

The goal of the workshop is to help people participate in this effort to bring even more Ada software to Debian, or to help maintain the existing packages.

What you should bring with you:

- your computer, already installed with Debian unstable or with an unstable chroot already created (see below);
- network cables (or WiFi already configured);
- monitor and keyboard, if your computer is not a laptop;
- power cables;
- some Ada software you would like to see in Debian but is not there (not necessarily software that you wrote; any software with a license permitting redistribution in source and binary form will do).

Note 1: if your computer does not run Debian as its main operating system, you can install Debian in a virtual machine (VMWare or other), in a jail on a FreeBSD system (Debian kfreebsd-i386), or in a chroot on any other distribution. Danny Beullens will offer help and assistance to those who would like to install Debian in a VMWare virtual machine.

Note 2: if you would like to install Debian as your main operating system but are uncomfortable doing so by yourself, please get in touch with your nearest Linux User Group (e.g. http://www.bxlug.be in Brussels).

What Ludovic Brenta will do for you:

- set up a local Debian mirror, so you can install or upgrade packages necessary for Ada package development;
- explain how to package Ada software for Debian;
- help you package your own program or library;
- answer questions about GNAT, GCC, Debian, etc.;
- if your package is suitable for inclusion in Debian, sponsor it for you.

What you will go home with:

- your own .deb packages installed on your computer;
- better understanding of how packaging works;
- better understanding of the Debian Policy for Ada;
- if your package is suitable, your name on the Debian Package Tracking System and your package on the next Debian DVD or CDROM distribution.

[7] http://www.debian.org/

[8] http://people.debian.org/~lbrenta/debian-ada-policy.html

Directions

To permit this more interactive and social format, the event takes place at private premises in Leuven. As instructed above, please inform us by e-mail if you would like to attend, and we'll provide you precise directions to the place of the meeting. Obviously, the number of participants we can accommodate is not unlimited, so don't delay...

Looking forward to meet many of you in Leuven!

Dirk Craeynest, President Ada-Belgium

Dirk.Craeynest@cs.kuleuven.be

Acknowledgments

We would like to thank our sponsors for their continued support of our activities: AdaCore, Katholieke Universiteit Leuven (K.U.Leuven), Offis nv/sa - Aubay Group, and Université Libre de Bruxelles (U.L.B.).

Ada-Europe 2010 — Final Call for Participation

From: Dirk Craeynest <dirk@asgard.cs.kuleuven.be>

Date: Sun, 6 Jun 2010 23:12:42 +0200 CEST

Subject: Press Release - Reliable Software Technologies, Ada-Europe 2010

Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

FINAL Call for Participation

*** UPDATED Program Summary ***

15th International Conference on

Reliable Software Technologies - Ada-Europe 2010

14 - 18 June 2010, Valencia, Spain

http://www.ada-europe.org/conference2010

*** Final Program available on conference web site. ***

*** Check out the tutorial program! ***

*** Printed proceedings available. ***

*** Register now! ***

Press release:

Ada-Europe Conference on Reliable Software Technologies

International experts meet in Valencia

Valencia (6 June 2010 23:00) - Ada-Europe, in cooperation with ACM's Special Interest Group on Ada, organizes the "15th International Conference on Reliable Software Technologies - Ada-Europe 2010" from 14 to 18 June in Valencia, Spain.

The conference offers two days of tutorials, three invited speakers, a full technical program of refereed papers, a collection of industrial presentations, a special session on Software Vulnerabilities and Security, an industrial exhibition, and a social program.

The 8 excellent tutorials on Monday and Friday cover a broad range of topics: Developing High-Integrity Systems with GNATforLEON/ORK+; Software Design Concepts and Pitfalls; Using Object-Oriented Technologies in Secure Systems; Hypervisor Technology for Building Safety-Critical Systems: XtratuM; How to Optimize Reliable Software; Developing Web-Aware Applications in Ada with AWS; SPARK: The Libre Language and Toolset for High-Assurance Software; C#, .NET and Ada: Keeping the Faith in a Language-Agnostic Environment.

Three eminent keynote speakers have been selected to open each day of the core conference program. 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 for Reliable Real-Time Systems?". Pedro Albertos (Universidad Politécnica de Valencia, Spain), a most authoritative member of the Automatic Control community will explore the relationship between implementation and performance of control algorithms in a talk entitled "Control Co-Design: Algorithms and their Implementation". James Sutton (Lockheed Martin, USA), a renowned expert software architect, in his talk entitled "Ada: Made for the 3.0 World" will explore how Ada is prepared for a world that makes peace with complexity and chaos, and learns to use them to its advantage.

The technical program presents 17 refereed and carefully selected papers on the latest research, new tools, applications and industrial practice and experience, a collection of 11 industrial presentations reflecting current practice and challenges, and an invited session on the hot topic of software vulnerabilities and security. Springer Verlag publishes the proceedings of the conference, as LNCS Vol. 6106.

The exhibition opens in the mid-morning break on Tuesday and runs continuously until the end of the afternoon break on Thursday. The exhibitors include the following vendors: AdaCore, Altran Praxis, Atego (formerly Aonix), and Ellidiss Software (formerly TNI Europe).

The social program includes on Tuesday evening a welcome reception at the Jardí Botànic, the botanical garden within walking distance from the conference venue, and on Wednesday evening a bus trip to and conference banquet in the Masía Xamandreu, a beautiful Valencian country house built in the 19th century.

The venue for the Ada-Europe 2010 conference is the Fundación Universidad-Empresa - ADEIT. It is a modern building located behind Santa Catalina Church, in the historical city centre of Valencia and very close to most historic buildings and monuments. The full program is available on the conference web site. Registration is still open.

Latest updates:

- The 12-page "Final Program" is available on the conference web site <http://www.ada-europe.org/conference2010>, and directly at <http://www.grupodicom.com/ae2010/AE-2010%20Final%20Program.pdf>.
- Check out the 8 tutorials in the final program (PDF) or via the hyperlinks in the tutorial schedule on <http://www.grupodicom.com/ae2010/conferenceprogram.html>.
- The proceedings, published by Springer Verlag as Lecture Notes in Computer Science Vol. 6106, are ready and will be distributed at the conference. More info is available at <http://www.springeronline.com/978-3-642-13549-1>.
- Registration fees are very reasonable and the registration can be done on-line (preferred) or by faxing a filled-out form to the conference secretariat. For all details, see <http://www.grupodicom.com/ae2010/registration.html>. Don't delay!
- For the latest information consult the conference web site.

Please circulate widely.

[…]

Review of Ada Issues

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Date: Tue, 4 May 2010, 07:53 GMT
Subject: Review of Ada Issues for June 2010 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 Ada-Europe 2010 conference next June in Valencia, Spain.

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>.

AI05-0031-1/03 2010-04-05 - Add a From parameter to Find_Token

AI05-0049-1/03 2010-04-06 - Extend file name processing in Ada.Directories

AI05-0113-1/05 2010-04-02 - Conflicting external tags and other tag issues

AI05-0124-1/02 2010-04-02 - Where is the elaboration check suppressed?

AI05-0136-1/06 2010-01-15 - Multiway tree container

AI05-0143-1/03 2009-06-27 - In Out parameters for functions

AI05-0144-2/06 2010-05-03 - Detecting dangerous order dependences

AI05-0149-1/06 2009-12-17 - Access types conversion and membership

AI05-0150-1/03 2010-02-04 - Use all type clause

AI05-0157-1/03 2009-12-10 - Calling Unchecked_Deallocation is illegal for zero-sized pools

AI05-0160-1/02 2009-12-17 - Additional ways to invalidate cursors

AI05-0162-1/03 2010-04-02 - Allow incomplete types to be completed by partial views

AI05-0164-1/02 2009-11-30 - Parameters of access-to-subprogram parameters and derivation

AI05-0166-1/06 2010-04-22 - Yield for non-preemptive dispatching

AI05-0168-1/04 2010-04-05 - Extended suspension objects

AI05-0176-1/06 2010-04-12 - Quantified expressions

AI05-0178-1/02 2009-12-11 - Incomplete views are limited

AI05-0179-1/04 2010-04-05 - Labels at end of a sequence_of_statements

AI05-0181-1/02 2009-12-12 - Soft hyphen is a nongraphic character

AI05-0193-1/02 2009-11-30 - Alignment of allocators

AI05-0196-1/02 2010-02-04 - Null exclusion checks for 'out' parameters

AI05-0203-1/02 2010-04-02 - A return_subtype_indication cannot denote an abstract subtype

AI05-0205-1/02 2010-04-02 - An extended return statement declares a name usable inside the statement

AI05-0207-1/02 2010-04-02 - Access constant is considered for mode conformance

AI05-0208-1/04 2010-04-16 - Characteristics of incomplete views

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, Saturday, May 29th, 2010. 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 30-4 (Dec 2009), p.207 —mp]

Ada and Education

Webinar on SPARK Pro 9

From: AdaCore Press Center
Date: Wed, 24 Mar 2010
Subject: SPARK Pro 9 now available
URL: http://www.adacore.com/2010/03/24/spark-pro-9/

[…]

A webinar providing an introduction and demonstration of SPARK Pro 9 will be held on April 27, 2010. It will begin at 5pm European Daylight Time/4pm GMT Daylight Time/11am Eastern Daylight Time/8am Pacific Daylight Time. To register please visit: http://www.adacore.com/home/products/gnatpro/webinars/

[…]

Ada-related Resources

Tutorials for AUnit

From: Jamie Ayre
Date: Tue, 11 May 2010
Subject: Daniel Bigelow's AUnit tutorials
Source: LinkedIn - Ada Programming Language

Daniel has kindly produced a number of tutorials for the Ada unit testing framework:

Part 1: Framework Overview

Part 2: Simple Test-Case Class Tutorial

Part 3: Standard Test-Case Tutorial

Part 4: Fixture Test-Case Tutorial

Part 5: Liskov Substitution Principle: 1 of 2

Part 5: Liskov Substitution Principle: 2 of 2

Part 6: Testing a class hierarchy using the Standard Test-Case: 1 of 2

Part 6: Testing a class hierarchy using the Standard Test-Case: 2 of 2

Part 7: Testing Generic Units: 1 of 2

Part 7: Testing Generic Units: 2 of 2

To view these, please visit:

http://www.adacore.com/2010/05/11/aunit-tutorials

Ada Reference Manuals in info format

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 21 May 2010 07:33:34 -0400
Subject: info version of Ada reference manuals
Newsgroups: comp.lang.ada

I've published an updated version of the Ada reference manuals in info format.

No change in the content; just fixed some info bugs.

http://www.stephe-leake.org/ada/arm.html

I'm working on packaging this for Debian.

Books on data structures

From: aprogrammer@nospam.org
Date: Fri, 14 May 2010 08:23:27 +0000
Subject: Good book(s) on data structures?
Newsgroups: comp.lang.ada

Hi,

Can anyone recommend a good book or books on data structures? I couldn't find an obvious newsgroup to post in, but given Ada and Ada people have a software-engineering approach lacking in many other communities I figured to ask here.

I understand and have implemented things like stacks, queues, and linked-lists in various projects. I never learned about data structures such as trees, and I'm sure there are many more I don't know about. I'm not interested in theory for the sake of theory, since I'm a practising software designer. What I am interested in is having the right kit of tools so I can apply the correct solution to the job. So I need good, practical sources rather than mathematical. I'm concerned about things such as clarity and performance in the code I write.

I prefer to stay away from books choosing this or that programming language as the lexicon, although books in Ada would be acceptable because of Ada's clarity. Thanks for any and all suggestions. […]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 17 May 2010 03:09:35 -0700 PDT
Subject: Re: Good book(s) on data structures?
Newsgroups: comp.lang.ada

I think you've come to the right place. I read several articles on data structures on Wikipedia and liked them a lot. They include just enough theory to help you choose which data structure to use and sample implementations in various languages. You can start browsing here:

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

the List of Data structures is particularly useful.

From: Phil Thornley <phil.jpthornley@googlemail.com>
Date: Fri, 14 May 2010 04:22:23 -0700 PDT
Subject: Re: Good book(s) on data structures?
Newsgroups: comp.lang.ada

[…]

The most obvious one referencing Ada is Mike Feldman's "Software Construction and Data Structures with Ada95".

ISBN 0-201-88795-9. It still seems to be in print.

[…]

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Fri, 14 May 2010 05:22:40 -0700 PDT
Subject: Re: Good book(s) on data structures?
Newsgroups: comp.lang.ada

[…]

Although neither is specific to Ada,

Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein

and

Algorithms + Data Structures = Programs by Wirth

are both excellent. CLRS's is positively encyclopedic, while Wirth's is an all time classic.

Both available via Amazon, eBay or even book shops!!

From: Colin Paul Gloster <Colin_Paul_Gloster@acm.org>
Date: Mon, 17 May 2010 10:47:33 +0000
Subject: Re: Good book(s) on data structures?
Newsgroups: comp.lang.ada

[…]

A newer version was available for free on Wirth's website. I have not read any version thereof.

Though it had a fake language based on Java without exceptions, the third edition of "Computer Algorithms: Introduction to Design & Analysis" by Sara Baase and Allen Van Gelder, published by Addison Wesley, was very good.

From: Mark Lorenzen <mark.lorenzen@gmail.com>
Date: Mon, 17 May 2010 07:35:11 -0700 PDT
Subject: Re: Good book(s) on data structures?
Newsgroups: comp.lang.ada

[…]

I recommend the following two books:

"Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein

http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11866

"Purely Functional Data Structures" by Chris Okasaki

http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521631242

From: Tom Moran <tmoran@acm.org>
Date: Mon, 17 May 2010 16:21:22 +0000 UTC

Subject: Re: Good book(s) on data structures?
Newsgroups: comp.lang.ada

[…]

"Software Components with Ada" by Grady Booch has a nice set of chapters on various data structures. The first section of each one has "The Abstraction; Constructors; Selectors; Iterators; Imports; Exceptions; Forms" and each ends with "Analysis of Time and Space Complexity".

Ada-related Tools

Simple components for Ada v3.8

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 9 Apr 2010 18:14:07 +0200
Subject: ANN: Simple components for Ada v3.8
Newsgroups: comp.lang.ada

The current version provides implementations of smart pointers, sets, maps, directed graphs, directed weighted graphs, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support; strings editing and tables management.

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

This release provides a persistence layer backed by SQLite. As a by-product it includes SQLite bindings. Differently to renown GNADE the bindings link the DB engine statically, which is probably the only case when one might wish to use SQLite.

Another enhancement is that the parser tools now support sources based on Ada streams. The stream is read using Character'Read. User-defined delimiters (like line ends) are supported.

From: Michael Rohan <michael@zanyblue.com>
Date: Fri, 9 Apr 2010 11:58:30 -0700 PDT
Subject: Re: ANN: Simple components for Ada v3.8
Newsgroups: comp.lang.ada

[…]

In another thread, it was pointed out that the standard Ada.Containers types are not safe in a multi-tasking environment. Since you mention locking wrt your structures, I assume they are safe in a multi-tasking environment?

[…]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 9 Apr 2010 21:20:54 +0200
Subject: Re: ANN: Simple components for Ada v3.8
Newsgroups: comp.lang.ada

[…]

Lock-free structures are safe when used as described on the page.

General case containers are not safe for the reasons same as for Ada.Containers.

[see also "Simple components for Ada v3.7" in AUJ 31-1 (Mar 2010), p.8; read also the thread mentioned above: "On the status of AdaCL and synchronization of containers in Ada" in AUJ 31-1 (Mar 2010), p.8 —mp]

GtkAda Contributions v2.6

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 10 Apr 2010 16:54:47 +0200
Subject: ANN: GtkAda Contributions v2.6
Newsgroups: comp.lang.ada

The software is 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-columned 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.

The present version provides minor bug fixes and some extensions to debugging support (GNAT based).

[see also "GtkAda Contributions v2.5" in AUJ 30-4 (Dec 2009), p.207 —mp]

Units of measurement for Ada v3.0

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 12 Apr 2010 13:45:09 +0200
Subject: ANN: Units of measurement for Ada v3.0
Newsgroups: comp.lang.ada

The library provides tools for handling dimensioned values in Ada. Checks are run-time when not removed by the compiler. String I/O and GTK+ widgets based on GtkAda included.

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

Changes to the version 2.9:

1. Output of exact zero values uses power 1 with any small. E.g. 0W is output as 0•W rather than 0•yW.

[see also "Units of measurement for Ada" in AUJ 29-3 (Sep 2008), p.152 —mp]

Interval Arithmetic for Ada v1.8

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 16 Apr 2010 12:16:21 +0200
Subject: ANN: Interval Arithmetic for Ada v1.8
Newsgroups: comp.lang.ada

The software provides an implementation of integer, float and dimensioned intervals.

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

In this version rounding behavior was changed in the case when a boundary is exact zero. It is left as is, even if the machine rounds. Because rounding cannot change the sign.

[see also "Interval arithmetic" in AUJ 29-3 (Sep 2008), p.152 —mp]

RAPID 3.3

From: Oliver Kellogg <okellogg@users.sourceforge.net>
Date: Wed, 12 May 2010 06:54:12 +0200
Subject: RAPID 3.3 is released
Newsgroups: comp.lang.ada

A new version of the Rapid Ada Portable Interface Designer is available at https://savannah.nongnu.org/files/?group=rapid (rapid-3.3.tar.gz)

The main features are:

- Added support for giving multiple GUI file names on the command line when in non-interactive mode (-ni)
- In non-interactive mode, new command line switch -od lets user choose different output directory for code generation
- New window feature "Snap to Grid" facilitates aligning widgets
- MinGW/MSYS rapid.exe contributed by Stefano Lagrasta

For more information, see the ChangeLog file or:

http://www.nongnu.org/rapid/docs/rapid_users_guide.html#features3

[see also "RAPID 3.2" in AUJ 30-2 (June 2009), p.74 —mp]

Ada-Python binding

From: Maciej Sobczak <maciej@msobczak.com>
Date: Fri, 7 May 2010 14:43:04 -0700 PDT
Subject: Ada-Python binding
Newsgroups: comp.lang.ada

[…]

The following article might be of interest to programmers practicing mixed-language development:

http://www.inspirel.com/articles/Ada_Python_Binding.html

This article builds on the various bits and pieces of knowledge that I needed to collect in order to proceed with one of my projects.

Hopefully it will be useful for others as well.

All comments are welcome.

From: Ludovic Brenta <ludovic@ludovic-brenta.org
Date: Sat, 08 May 2010 00:24:11 +0200
Subject: Re: Ada-Python binding
Newsgroups: comp.lang.ada

[…]

The GNAT Programming Studio uses this technique extensively. Indeed, Python is not only the scripting language for GPS but also the language in which parts of GPS are written. GPS exposes a rich API to the Python scripts and plug-ins.

You can have a look at the relevant part of the GPS sources here:

[1] http://libre2.adacore.com/viewvc/trunk/gps/gnatlib/src/python

[2] http://libre2.adacore.com/viewvc/trunk/gps/python

[3] http://libre2.adacore.com/viewvc/trunk/gps/share/library

[4] http://libre2.adacore.com/viewvc/trunk/gps/share/plug-ins

[1] is part of what AdaCore want to turn into a reusable library; currently the sources are still "hidden" inside the GPS source tree. This is, essentially, an Ada-Python binding that allows not only to call Ada subprograms from Python but also to launch a Python interpreter and execute scripts from within an Ada program.

[2] loads the Python interpreter into GPS for to execute plug-ins and user-defined initialization scripts written in Python.

[3] contains a library of reusable components written in Python, for use in plug-ins or user-defined scripts.

[4] contains the plug-ins written in Python.

Most of the modules of GPS export an API to these plug-ins; the code that does that is scattered in the rest of the source tree.

From: Cyrille Comar <comar@eu.adacore.com>
Date: Mon, 10 May 2010 02:09:17 -0700 PDT
Subject: Re: Ada-Python binding
Newsgroups: comp.lang.ada

[…]

you also have on-line documentation at

http://libre.adacore.com/wp-content/files/auto_update/gnatcoll-docs/gnatcoll.html

Ada binding for ØMQ

From: Thomas Løcke
Date: Sun, 23 May 2010
Subject: ØMQ Ada binding for the masses
URL: http://ada-dk.org/?page=news&news_id=149

Posting this news item is actually a bit of a blast from the past.

Way back when I started doing web development, I used the excellent webserver Xitami, created by iMatix.

It served me well for several years, and as far as I know, it's once again in active development.

From the same company comes ØMQ, a messaging system that is fast, scalable and simple to use:

This little library is really fast, and you can learn to use it in about one hour.

ØMQ does the hard work of messaging behind the scenes. It's portable, runs on lots of operating systems and has loads of language interfaces. And your apps stretch to many cores, and many boxes, like magic. Connect your application threads and processes together using message patterns like request-reply and publisher/subscriber. Patterns are building blocks you can use individually, or connect in many ways. You access ØMQ using a simple socket API. From any language and on any OS (almost).

Ada programmers will be glad to know that ØMQ comes with a complete Ada binding, which appears to be very easy to use.

My experience with iMatix software is that it is performant and stable, so if you need a messaging system, this just might be it.

[find the ØMQ website at http://www.zeromq.org —mp]

Ada binding for AMQP

From: Grupo de Tecnología Informática-Inteligencia Artificial - Universidad Politécnica de Valencia
Date: May 2010 [fetched]
Subject: Ada Binding to AMQP
URL: http://users.dsic.upv.es/grupos/ia/sma/tools/AdaBinding/index.php

The Advanced Message Queuing Protocol (AMQP) is an open standard for an interoperable asynchronous messaging protocol. The Ada Binding to AMQP allows applications built on top of it to be distributed over heterogeneous environments. In this sense, it increases the interoperability, portability, and flexibility of such applications. In this paper, we propose an Ada binding to AMQP. This binding has been implemented using Apache Qpid, an open-source implementation of the AMQP standard.

[Verbatim from the webpage of the project (end of May 2010). The Ada binding is available for download at: http://users.dsic.upv.es/grupos/ia/sma/tools/AdaBinding/downloads.php —mp]

On database interface libraries for Ada

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 29 Apr 2010 06:41:32 -0700 PDT
Subject: On database interface libraries for Ada
Newsgroups: comp.lang.ada

A recent discussion on the AWS users' mailing list made me realize that there exist many, many different libraries for interfacing Ada programs to various database engines. I even learned about a couple that I was not yet aware of.

I think it would be very nice to consolidate all the information available on all these libraries in one place, so that programmers know about them and can choose wisely. It seems to me that the best place for such a discussion is the existing page on the "Ada Programming" wikibook:

http://en.wikibooks.org/wiki/Ada_Programming/Libraries/Database

This page was very incomplete, listing only GNADE, APQ and GWindows with links to their home pages but no other information. I have edited this page to provide more information in a table. The information there is still incomplete; please add your favourite database interface library to the page for better visibility and correct the information as you deem necessary. Also feel free to add notes and details outside the table.

Thanks for your help.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 30 Apr 2010 12:08:01 +0200
Subject: Re: On database interface libraries for Ada
Newsgroups: comp.lang.ada

[…]

What about a separate section for embedded / single file data bases e.g. SQLite, Berkeley etc? From my point of view it is a quite different topic from the bindings to/as DBMS clients.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 30 Apr 2010 03:31:33 -0700 PDT
Subject: Re: On database interface libraries for Ada
Newsgroups: comp.lang.ada

[…]

On the one hand I think that's a valid distinction. On the other, such a split would lead to duplicate information since, for example, GNADE, QtAda and gnatcoll support both embedded and client-server databases. Also, this is a property of the target database engine, not necessarily of the Ada binding. Maybe adding a note that SQLite is in-process (linked either statically or dynamically) would be sufficient. (I do not know of a Berkeley DB binding, yet...)

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 30 Apr 2010 14:47:55 +0200
Subject: Re: On database interface libraries for Ada
Newsgroups: comp.lang.ada

[…]

> (I do not know of a Berkeley DB binding, yet…)

I remember that somebody announced it. (Before I did SQLite, I tried to find any information about these bindings, but failed. Then I considered to do it by myself, but was horrified by its interface. So in the end I decided in favor of SQLite.)

From: Xavier Grave <xavier.grave@ipno.in2p3.fr>
Date: Fri, 30 Apr 2010 08:48:10 +0200
Subject: Re: On database interface libraries for Ada
Newsgroups: comp.lang.ada

[…]

Thanks Ludovic for this URL and the information it contains. I'm working with sqlite3 now, I'll be glad to have a more complete binding to it using SQLite3-Ada than the very poor one I have developed for my needs (it only uses blob).

[…]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 30 Apr 2010 09:27:41 +0200
Subject: Re: On database interface libraries for Ada
Newsgroups: comp.lang.ada

[…]

I have SQLite3 bindings (middle thickness) in the Simple Components:

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

It is more than blobs, but not all SQLite.

P.S. GNADE has SQLite bindings as well. I didn't use them because I wanted to be able to put the amalgamation directly into the project, the only sensible use of SQLite, IMO.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 30 Apr 2010 02:28:44 -0700 PDT
Subject: Re: On database interface libraries for Ada
Newsgroups: comp.lang.ada

[…]

It would be nice if you would add the relevant information to the wiki page for future reference. This should not take you more time than writing your post did. Also explaining the differences between the 4 different bindings to SQLite (5 with yours) would help.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 29 Apr 2010 06:51:58 -0700 PDT
Subject: Re: On database interface libraries for Ada
Newsgroups: comp.lang.ada

[…]

Side note: since my changes have not yet been "sighted" (whatever that means), you must click on "View draft" on the top-right corner of the page to see the most recent version.

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Thu, 29 Apr 2010 17:22:01 +0300
Subject: Re: On database interface libraries for Ada
Newsgroups: comp.lang.ada

[…]

Another place which lists various database bindings/libraries is AdaCommons wiki:

http://www.adacommons.org/Category:Database

[…]

VC_View 2.1.1

From: Phil Thornley <phil.jpthornley@googlemail.com>
Date: Tue, 1 Jun 2010 09:59:56 -0700 PDT
Subject: ANN: VC_View 2.1.1 is Linux compatible
Newsgroups: comp.lang.ada

VC_View presents verification conditions, generated by the SPARK tools, in a way that makes them easier to interpret:

1. Only immediately relevant hypotheses are initially displayed.
2. User identifiers are replaced by upper-case letters.

Version 2.1.1 of VC_View is now available and is compatible with both Windows and Linux.

The source distribution (GPL) and a Windows executable are available on the Download page at www.sparksure.com.

(The program functionality is the same as the previous Windows only version, 2.0).

Many thanks to Alexander Senier for sorting out the compatibility problems.

[see also "SPARK Proof" in AUJ 30-2 (Jun 2009), p.73 —mp]

Matreshka v. 0.0.3

From: Vadim Godunko <vgodunko@gmail.com>
Date: Sun, 11 Apr 2010 13:18:00 -0700 PDT
Subject: Announce: Matreshka 0.0.3
Newsgroups: comp.lang.ada

New version of Matreshka was released. Previous version can't be used on 32-bit platforms, this one fixes this. Also, new version includes:

- automatic configuration facility; it detects host architecture and compiler's capabilities to select most suitable of supported configurations;
- upgrade of Unicode Character Database to version 5.2.0;
- upgrade of Unicode Collation Algorithm to version 5.2.0;
- upgrade of regular expression engine to support character's properties from new version of Unicode;
- bug fixes.

Source code and documentation can be found on project's website

http://adaforge.qtada.com/cgi-bin/tracker.fcgi/matreshka/

[see also "L10n and i18n support in Ada" in AUJ 30-3 (Sep 2009), p.147 —mp]

TXL Grammar for Ada 2005

From: Bruno Le Hyaric <bruno.lehyaric@gmail.com>
Date: Mon, 17 May 2010 15:18:37 -0700 PDT
Subject: TXL Grammar for Ada 2005
Newsgroups: comp.lang.ada

[…]

I've recently made up a new grammar for Ada based on the great language manipulation tool: TXL (http://www.txl.ca)

http://github.com/bu2/ ada-2005-txl-grammar

The grammar is currently at an early stage. It permits to parse full Ada 2005 following LRM standard but still only provides parsing trees.

AST coming soon and even more...

Comments are welcome!

Ada-related Products

AdaCore / Altran Praxis — SPARK Pro 9

From: AdaCore Press Center
Date: Wed, 24 Mar 2010
Subject: SPARK Pro 9 now available
URL: http://www.adacore.com/2010/03/24/ spark-pro-9/

New SPARK Pro 9 Development Environment Delivers Increased Security Assurance for Critical Projects

AMSTERDAM, NEW YORK and PARIS, March 24, 2010 – Avionics Europe 2010 – SPARK Pro 9, announced today by AdaCore and Altran Praxis, provides a major step forward for developers creating safety critical and high assurance systems. The advanced open source development environment now features increased security functionality, including the ability to verify and assure Multiple Independent Levels of Security (MILS) within the same application as well as support for the latest SPARK2005 language profile.

SPARK Pro was created one year ago by Altran Praxis, international specialist in embedded and critical systems engineering, and AdaCore, the leading provider of commercial software solutions for the Ada language. SPARK Pro provides the foremost language, toolset and design discipline for engineering high-assurance software. It combines Altran Praxis' renowned SPARK language and verification tools, with the GNAT Programming Studio (GPS) and GNATbench development environments from AdaCore.

The new release of SPARK Pro 9 demonstrates the forward momentum of the toolset and introduces significant new features to benefit developers. Chief amongst these is the ability to mix software of different security levels (such as classified and unclassified) within the same system. This MILS functionality meets the increasing trend in the aerospace and defence industries to combine multiple secure and non-secure elements into a single system to deliver smaller, more integrated solutions.

"We've seen significant growth in the use of SPARK Pro since its launch last year across the safety critical and high assurance markets," said Keith Williams, Altran Praxis Managing Director. "SPARK Pro 9 continues this market momentum and introduces important new features, particularly in the growing high security sector. It demonstrates the continuing benefits of our close partnership with AdaCore to our customers across the globe."

Over half of new features in SPARK Pro 9 are the result of customer feedback. As well as support for the advanced features of SPARK2005, the latest version of the SPARK language, closer integration between SPARK and GPS results in improved usability and faster development times.

"In MILS-oriented Operating Systems, mixing safely different levels of security assurance requires a complex multi-partition organization. SPARK Pro 9 now provides the means to verify accurately such a mix in the context of a single application through sound information flow analysis," said Cyrille Comar, Managing Director, AdaCore. "This capability offers an unprecedented level of flexibility for those writing rich applications with stringent security requirements."

Developed by Praxis, SPARK is a language specifically designed to support the development of software used in applications where correct operation is vital for reasons of safety, security, or both. 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. There are versions of SPARK based on Ada 83, Ada 95, and Ada 2005, so all standard Ada compilers and tools work out-of-the-box with SPARK.

SPARK Pro 9 new functions include:

- New information-flow verification for safety and security policies, such as Bell-LaPadula, based on integrity labelling of variables, inputs and outputs. This facility allows users to confirm intended separation properties, and to prevent violations of the chosen information flow policy.
- SPARK2005. The new SPARK2005 language profile is now available. At present, Ada2005 features supported include 'Mod, 'Machine_Rounding, new reserved words, and the static semantics of "overriding."
- New ZombieScope tool, which detects dead statements, branches and paths in SPARK code, complementing the capabilities of the Simplifier and proof status summarizer POGS.
- Cross Referencing annotations in GPS. The Examiner now generates cross-reference information that can be consumed by GPS to drive navigation within annotations.
- Function return annotations are now treated more like procedure post-conditions, being substituted into the VC hypotheses of the caller. This can dramatically improve the effectiveness of the theorem prover for those calling units, as well as reducing the manual work required by the user to provide rewrite rules.
- New output format for POGS. This format is designed to be both easier to read and easier to search automatically. It also reflects the results of the new ZombieScope tool.
- Simpler documentation structure. All proof material is now in one manual, and a new global index (in both clickable PDF and HTML forms) simplifies finding topics in the entire manual set.
- Case checking. New Examiner switch that insists on consistent casing within annotations.

[…]

About Altran Praxis

Altran Praxis is a specialist systems and software house, focused on the engineering of systems with demanding safety, security or innovation requirements. Altran Praxis leads the world in specific areas of advanced systems engineering and innovation such as: ultra low defect software engineering, Human Machine Interface (HMI), safety engineering for complex or novel systems and tools (such as SPARK) /methods for systems engineering. It offers clients a range of services including turnkey systems development, consultancy, training and R&D.

Key market sectors are aerospace and defence, rail, nuclear, air traffic management, automotive, medical and security. The company operates globally with active projects in the US, Asia and Europe. The headquarters of Altran Praxis are in Bath (UK) with offices in Sophia Antipolis, London, Paris, Loughborough and Bangalore. Altran Praxis is an expertise centre within, and wholly owned by, Altran which is a global leader in innovation engineering and employs 17,000 staff across the world.

www.altran-praxis.com

About AdaCore

Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, the 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 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, railway systems and medical devices, and in security-sensitive domains such as financial services.

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

AdaCore — GNAT Pro High-Integrity Edition for VxWorks MILS Platform

From: AdaCore Press Center
Date: Tue, 27 Apr 2010
Subject: GNAT Pro High-Integrity Edition For VxWorks MILS Platform now available
URL: http://www.adacore.com/2010/04/27/mils/

GNAT Pro High-Integrity Edition For VxWorks MILS Platform now available

A complete security application development environment for EALs 1 through 7

NEW YORK, PARIS and SAN JOSE, Calif., April 27, 2010 – Embedded Systems Conference – AdaCore, a leading supplier of Ada development tools and support services, today announced the first major release of its GNAT Pro High-Integrity Edition for MILS. The High-Integrity Edition for MILS (Multiple Independent Levels of Security) product is a specialized security application development environment supporting the creation and security certification of applications at the high end for EALs (Evaluation Assurance Levels) 5 through 7, as well as applications at the lower bounds for EALs 1 through 4. This product contains several specialized run-time libraries that support different levels of certification, specialized tools to support security certification, and, as an option, the SPARK language tool set to facilitate both development and certification of applications to top security levels.

GNAT Pro High-Integrity Edition for MILS contains the GNAT Pro development environment with compile system, testing tools, and graphical user interface components, to support high-productivity software development. It provides specialized run-time libraries, corresponding to Ada language subsets for those features that can be certified to different EALs. For EALs 1-3, the High-Integrity Edition for MILS provides a full Ada run-time library and development environment for the creation and testing required for these lower-level security certification requirements. For EAL 4, it provides a Ravenscar-compliant run-time library originally created to be certifiable to the DO-178B airborne avionics safety standard. This safety standard has been shown to meet the security assurance requirements for EAL 4. Finally, the Zero Foot Print (ZFP) run-time library is provided, which supports security certification to EALs 5 through 7.

Security certification at EALs 5-7 requires semi-formal or formal approaches. General software application languages are typically not appropriate at these levels. As an option, GNAT Pro High-Integrity Edition for MILS provides the SPARK Pro tool set to support these top levels. The SPARK language is a fully deterministic and verifiable subset of the Ada programming language augmented by pre- and post-condition constructs that more fully specify the application's logic and information flow. SPARK and the SPARK Pro tool set have been demonstrated in practice to meet the requirements for top security application development. The High-Integrity Edition for MILS thus provides a complete environment for applications that need to be certified at EALs 5-7.

"AdaCore has a long history in providing solutions for developers of high-integrity applications," said Robert Dewar, President and CEO of AdaCore. "GNAT Pro has been used for avionics systems on aircraft, such as the new Boeing 787, which need to meet the highest level of safety requirements in DO-178B. Developers of high-security applications require the more stringent security objectives be met. To satisfy this need, we have introduced the GNAT Pro High-Integrity Edition for MILS. This product, integrated with the SPARK Pro language and tool set, offers a unique security solution, allowing developers using a MILS architecture to create and certify applications that can meet all EALs from lowest to highest."

"Wind River VxWorks MILS platform provides a robust MILS architecture, enabling the creation of MLS (Multi-Level Secure) systems in which multiple applications from different domains execute securely on a single instance of silicon," said Chip Downing, Director of Aerospace and Defense at Wind River. "GNAT Pro High-Integrity Edition for MILS adds the capability to accelerate the certification of Ada applications at high assurance levels on the VxWorks MILS foundation for a very powerful combination for developing secure applications."

Availability

GNAT Pro High-Integrity Edition for MILS is available today for the VxWorks MILS Platform.

[…]

Adalog — AdaControl 1.12r3

From: J-P. Rosen <rosen@adalog.fr>
Date: Tue, 13 Apr 2010 10:50:01 +0200
Subject: AdaControl 1.12r3 released
Newsgroups: comp.lang.ada

Adalog is pleased to announce a new release of AdaControl.

This release includes various improvements, bug fixes, etc. and raises the number of possible checks to 388.

As usual, it is available in source (GMGPL) and executable for GNAT-GPL2009 forms from http://www.adalog.fr/adacontrol2.htm

[…]

[see also "AdaLog — AdaControl 1.11r4" in AUJ 30-3 (Sep 2009), p.148 —mp]

Atego — ObjectAda Real-Time for VxWorks 653

From: Atego Press Center
Date: Tue, 2 Mar 2010
Subject: Aonix ObjectAda Real-Time released on VxWorks 653
URL: http://www.aonix.com/pr_03_02_10a.html

Atego's real-time Ada development system now available to military and aerospace developers

Embedded World 2010, Nürnberg, Germany – 2nd - 4th March 2010 – Hall 11, Stand 119 — Atego, the leading independent supplier of industrial-grade, modeling and development tools for complex, mission and safety-critical embedded systems and software, has announced the release of Aonix ObjectAda Real-Time for Windows targeting the Wind River VxWorks 653 multi-partition RTOS for PowerPC. This is the first release of the Aonix ObjectAda product with full Ada runtime support for the Wind River safety-critical platform.

"Aonix ObjectAda Real-Time for VxWorks 653 was developed to meet customer demand," said James B. Gambrell, Executive Chairman at Atego. "Although Atego, formerly Aonix, has for years been providing a full safety-critical Ada product for ARINC-653 systems, to realize the full potential of multi-partition RTOSs, both full Ada runtime, and safety-certifiable Ada runtimes needed to be available. We are happy to now have satisfied this need."

VxWorks 653 is Wind River's robust operating system for controlling complex ARINC 653 Integrated Modular Avionics (IMA) systems. VxWorks 653 fully implements the latest ARINC 653 application executive (APEX) specification with a robust time and space partition scheduler. Aonix ObjectAda Real-Time runs within a VxWorks 653 partition where execution of a non-certifiable Ada application can work in cooperation with safety-critical functionality resident in other partitions. This is a significant cost advantage via reduction in the amount of Ada code that needs to endure the high cost and time associated with the rigor of certification testing and acceptance cycles.

Atego's Aonix ObjectAda Real-Time joins with Aonix ObjectAda RAVEN as one of two products now supporting the VxWorks 653 platform.

"Support for multiple levels of safety criticality is an important aspect of IMA systems," said Adrian Larkham, Head of Critical Systems at Atego.

"Aonix ObjectAda provides facilities needed for the development of Ada applications at varying levels of safety criticality within an IMA system.

Aonix ObjectAda Real-Time for VxWorks 653 provides full Ada language coverage for development projects with low levels of application safety criticality.

Application development at the highest level of safety criticality is covered by ObjectAda RAVEN for VxWorks 653 and its certifiable subset of Ada."

Aonix ObjectAda Real-Time for Windows x PowerPC/VxWorks 653 consists of a fully compliant ACATS 2.5 Ada 95 compiler plus supporting tools. It is compatible with Wind River's VxWorks 653 environment, which comprises the VxWorks 653 operating system and the VxWorks 653 cross development toolset.

Aonix ObjectAda for VxWorks 653 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 Aonix ObjectAda's standard graphical or command-line interface. The Aonix 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.

About the Aonix ObjectAda Family

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

About Atego

AtegoTM is the leading independent supplier of industrial-grade, collaborative development tools for engineering complex, mission- and safety-critical architectures, systems, software and hardware. Atego delivers a stable, robust working environment to thousands of users across an extensive range of complex applications in demanding engineering sectors such as aerospace, defense, automotive, transportation, telecommunications, electronics, and medical. Atego's Aonix PERC® is the leading, highly reliable, real-time embedded virtual machine solution for running JavaTM programs deployed today. Atego also has the largest number of certified Ada applications (Aonix ObjectAda®) at the highest level of criticality. Atego's standards-based modeling tool suite, Artisan Studio®, provides comprehensive support for the leading industry standards, including OMG SysML, UML and Architectural Frameworks. Atego WorkbenchTM provides a fully integrated, collaborative engineering framework for the trouble-free deployment and maintenance of best-in-class tools for mission and safety-critical systems and software development. Atego's tools deliver on the promise of an integrated collaborative development environment - allowing architecture, systems, software and hardware engineering teams to Work as OneSM - from concept through to delivery, maintenance and support. Founded in 2010 in a merger between Artisan Software Tools and Aonix, Atego is headquartered in San Diego, CA, USA and Cheltenham, UK with offices in France, Germany and Italy, and is supported by a global distributor network.

For more information visit:

www.atego.com.

[…]

Atego — Artisan Studio 7.2

From: Atego Press Center
Date: Thu, 3 Jun 2010
Subject: Atego launches Artisan Studio 7.2
URL: http://www.aonix.com/pr_06_03_10.html

Major release of Artisan Studio introduces role-based Editions; completed DoDAF support for UPDM; Artisan Studio Reviewer enhanced to deliver 80 new metric reports, usability improvements to Artisan Publisher and a new model comparison tool.

San Diego, USA and Cheltenham, UK — 3rd June 2010.

Atego, the leading independent supplier of industrial-grade, collaborative development tools for engineering complex, mission- and safety-critical architectures, systems, software and hardware, has launched Artisan Studio 7.2, a major new version of its flagship model-driven development tool suite.

Artisan Studio 7.2 delivers a variety of significant new modeling capabilities and functions. Artisan Studio has been re-architected to provide role-based Editions which have been specifically designed to make the working environment more relevant and efficient for the specialist needs of Enterprise Architects, Systems Engineers and Software Engineers. Artisan Studio also adds DoDAF capabilities to complete its support for the OMG's UPDM 1.0 standard for defense architectural frameworks. Artisan Studio 7.2 also extends Artisan Studio Reviewer with metric reporting and simplifies the user interface to Artisan Publisher as well as continuing to improve core technologies with a new model comparison tool, the Artisan Model Differencer, and adds more functionality for the Automatic Code Synchronizer and Activity Modeling.

"The launch of Version 7.2 continues to extend the market-leading position of Artisan Studio with the introduction of significant new concepts, features and functions to improve the development of mission and safety-critical embedded systems and software development," said Hedley Apperly, Vice-President of Product Marketing at Atego. "The introduction of role-based Editions is particularly innovative, aligning Artisan Studio with the specific roles and needs of Enterprise Architects, Systems Engineers and Software Engineers, while allowing them all to continue to Work-as-One."

Artisan Studio Architect Enterprise Edition focuses on the capabilities required by an Enterprise Architect. It provides comprehensive support for UPDM including both DoDAF and MODAF, which addresses the visual modeling needs of the global defense community by providing support for all seven UPDM viewpoints and a standardized means to describe both DoDAF and MODAF architectures which delivers enormous benefits for developers of military systems in terms of design consistency, quality and efficiency, improved tool interoperability and cost savings.

Artisan Studio Designer Enterprise Edition will be the tool of choice for Systems Engineers. They can take advantage of Artisan Studio's SysML profile to build systems and systems of systems using the industry standard SysML concepts and modeling language. Artisan Studio allows users to focus on systems engineering by providing the capability to document requirements, analysis diagrams, traceability and manage change all within one tool using Artisan Studio's integration with leading 3rd party tools like DOORS and

Mathworks Simulink.

Artisan Studio Developer Enterprise Edition is for Software Engineers. It enables the generation of code from the design and is fully equipped with Artisan Studio's Automatic Code Synchronizer and Transformation Development Kit to allow them to automatically generate code from a design that matches the company's best practices and coding standards. This edition is available for the C/C++, C#, VB, Java and Ada programming languages.

The highly successful Artisan Studio Reviewer now has a whole new model metrics facility with 80 new metric reports, 13 user-defined reviews and new customer-requested reviews. The metrics provide powerful statistics for managers on the maturity of the models, complementing the correctness, completeness and consistency checks that were already available.

With the launch of Artisan Studio 7.2, Artisan Publisher now has facilities to copy elements directly from an open Artisan Studio model and clearly view the content of the document. Additionally, many new templates have been introduced, specifically for the quick creation of repeatable and configurable output in UPDM, SysML and UML designs.

Finally, Artisan Studio Version 7.2 includes code generators for POSIX compliant operating systems and many other lower level improvements to its modeling environment.

"The launch of Artisan Studio 7.2 continues to deliver on our Work-as-One strategy," said James B. Gambrell, Executive Chairman of Atego. "Not only does Artisan Studio provide a single, purpose-built tool suite that joins up embedded engineering disciplines, teams and organizations but, by introducing the concept of role-based Editions and metrics reporting, it also now specifically caters for the requirements of the diverse and disparate specialists needs in the design chain - Enterprise Architects, Systems,

Software Engineers and Managers - in ways that match their explicit job functions and needs."

Inspirel — YAMI4

From: Inspirel website
Date: May 2010 [fetched]
Subject: YAMI4 - Messaging Solution for Distributed Systems
URL: http://www.inspirel.com/yami4/

YAMI4 is a set of messaging libraries designed for distributed systems with particular focus on control and monitoring systems.

The major features of the YAMI4 library are:

- peer-to-peer messaging without dependency on external message brokers
- support for message priorities
- built-in support for load balancing and automatic fail-over
- support for simple publish-subscribe messaging
- support for isolated memory partitions (intended for embedded and critical systems)
- support for real-time development with comprehensive range of timeout features
- high performance and scalability with non-blocking I/O
- small memory and resource footprint
- no dependencies on other libraries

The YAMI4 libraries are available for Ada, C++ and Java and are supported on POSIX-compliant systems, Microsoft Windows and Java-based platforms.

[Verbatim from the webpage of the project (End of May 2010) —mp]

From: Inspirel News center
Date: Mon, 1 Mar 2010
Subject: YAMI4-1.0.0
URL: http://www.inspirel.com/news.html

The Inspirel website gets a new look.

YAMI4-1.0.0, a messaging solution for distributed systems, is released.

This release marks a beginning of a new development line for the YAMI project and is meant to address the needs of demanding users working with mission-critical distributed systems.

From: Inspirel News center
Date: Fri, 12 Mar 2010
Subject: YAMI4-1.0.1 is released
URL: http://www.inspirel.com/news.html

YAMI4-1.0.1 is released.

This release introduces corrections for 64-bit systems.

From: Inspirel News center
Date: Fri, 9 Apr 2010
Subject: YAMI4-1.0.2 is released
URL: http://www.inspirel.com/news.html

YAMI4-1.0.2 is released. This release introduces several bugfixes for timeout handling in Java and system limits correction for Mac OS X Snow Leopard systems.

Rapita Systems — RapiTime v2.3

From: Rapita Systems News center
Date: Thu, 20 May 2010
Subject: Step back in time with RapiTime
URL: http://www.rapitasystems.com/news/v2.3+launched

York, UK, 20th May 2010

Rapita Systems Ltd has unveiled the latest version of its RapiTime measurement and analysis of real-time embedded software tool.

Combining the proven data collection, measurement and analysis techniques of previous versions with new features means RapiTime v2.3 takes timing analysis of real-time embedded systems "to a higher level".

New features to support execution time measurement, code coverage and worst-case execution time analysis include:

- A specialised debugging facility which gives users the ability to move backwards and forwards - "Rewind" - through source code
- The opportunity to run coverage without timing analysis
- Improvements in how RapiTime handles the analysis of function pointers and RTOS tasks

These new features mean RapiTime v2.3 quickly highlights opportunities for debugging and code optimization, rapidly leading the way to improved systems performance.

"With RapiTime v2.3 we've added features that improve the user experience and significantly enhance systems performance," said Product Development Director, Dr Antoine Colin.

"We wanted to take analysis to a higher level," continued Dr Colin, "and we're confident RapiTime v2.3 meets that ambitious aim."

The Rewind facility at the heart of v2.3 gives users a tool they can use at their own pace to identify and debug problem code.

Too often embedded developers face the situation where the target just stops working. Stepping backwards from the point where the application stopped to see the sequence of events leading up to the problem helps developers to identify the cause.

Combined with the other key features of v2.3, Rewind offers an advanced route to performance optimisation.

RapiTime v2.3 is officially launched on 24th May 2010.

[For more details, see http://www.rapitasystems.com/system/files/ProductBrief_RapiTimev2.3.pdf —mp]

Ada and GNU/Linux

qtada, gprbuild and xmlada for Debian, Ubuntu and Gentoo Linux

From: Alexander <coopht@gmail.com>
Date: Wed, 28 Apr 2010 06:24:33 -0700 PDT
Subject: Announce: qtada,gprbuil and xmlada for Debian, Ubuntu and Gentoo Linux
Newsgroups: comp.lang.ada

[…]

I prepared QtAda (www.qtada.com) packages for Debian/Ubuntu and Gentoo Linux.

For Gentoo Linux I created a special Ada-overlay. In this overlay there are ebuild files for gnat-gcc-4.4.3, gprbuild, xmlada and qtada.

The packed overlay can be downloaded from here:

https://sourceforge.net/projects/qtada/files/Gentoo/ada-overlay.tar.bz2/download

For Debian and Ubuntu Linux I created .deb packages for gprbuild, xmlada and qtada.

gprbuild package:

https://sourceforge.net/projects/qtada/files/Debian/gprbuild_1.3.0-1_i386.deb/download

xmlada package:

https://sourceforge.net/projects/qtada/files/Debian/libxmlada-dev_4.3.1-1_i386.deb/download

qtada package:

https://sourceforge.net/projects/qtada/files/Debian/qtada-dev_3.0.0-1_i386.deb/download

P.S. Please, don't be confused with project name

References to Publications

Embedded.com — “Expressive vs. permissive languages: Is that the question?”

From: AdaCore Press Center
Date: Thu, 8 Apr 2010
Subject: Expressive vs. permissive languages: Is that the question?
URL: http://www.adacore.com/home/company/press-center/

[Yannick Moy at Embedded.com discusses how the expressiveness and permissiveness of programming languages influence the static analyzability of the code. He contends that the chosen language does matter when static analyzability enters the picture. Read the article at:

http://www.embedded.com/design/224200704 —mp]

Ada Inside

Use of Ada in the Bound-T timing analysis tool

From: Ada Information Clearinghouse
Date: Sat, 20 Mar 2010
Subject: Ada drives Bound-T
URL: http://www.adaic.com/atwork/bound-t.html

[…]

Bound-T is a novel program analysis tool that takes the object code representation of an embedded program and analyzes it to produce bounds on worst-case execution time and worst-case stack usage. It supports a wide variety of embedded target processors, and (since it processes object code), it supports virtually any programming language and development methodology.

Bound-T was developed completely in Ada other than a couple external mathematics packages.

Niklas Holsti, Lead Engineer at Tidorum Ltd. says "The Bound-T tool is meant to help the verification of high-reliability software. Thus it is both important and natural that Bound-T itself should use a programming language that emphasises reliability."

Holsti has found that many features of Ada help improve the reliability of a tool.

He notes "Ada makes a clear separation between the interface of a module, and the implementation of the module. For Bound-T, in particular, this helps to separate the general, target-independent modules from the target-specific modules."

With some 480 packages, reduced coupling is critical to managing the complexity of Bound-T.

Ada's checks that both uses and implementations of a module match the interface of the module prevent many errors and reduce debugging time.

Bound-T also makes use of object-oriented programming as embodied by Ada's tagged types to reduce dependencies between modules while not hampering further development and refinement of the modules.

Holsti summarizes "Ada is the outstanding choice for reliability, through its strong support for clear, problem-oriented design and its thorough consistency checks at compile-time and at run-time."

Hi-Lite Project

From: AdaCore Press Center
Date: Tue, 4 May 2010
Subject: AdaCore Announces Hi-Lite Project
URL: http://www.adacore.com/2010/05/04/hi-lite-project/

New consortium aims to promote formal methods for high integrity software

PARIS and NEW YORK, May 4, 2010 – AdaCore, together with Altran Praxis, CEA LIST, Astrium Space Transportation, INRIA ProVal and Thales Communications today announced the start of the Hi-Lite project. Financially supported by French national and local government agencies, Hi-Lite is an Open Source project designed to increase the use of formal methods in developing high integrity software, particularly to meet the forthcoming DO-178C avionics standard. It will achieve this by building tools that are simpler, more powerful and easier to use.

Hi-Lite will bring together the strengths of the project partners to create formal verification tools for both the Ada and C languages. These will enable code verification at a deeper level than current solutions and reduce the need for time-consuming and costly physical testing of high integrity software solutions. The €3.9 million ($5.3 million) project is scheduled to last three years.

The project builds on the existing ten year experience of Airbus in using formal verification methods to create high integrity systems, and is strongly driven by the criteria that Airbus’s work has generated: soundness, applicability to the code, usability by “normal” engineers on “normal” computers, improvement on classical methods, and certifiability.

The project is structured as two different tool chains for Ada and C.

AdaCore will lead the project and contribute its expertise in the Ada language, including the GNAT compiler and CodePeer static analyser, with Altran Praxis providing its Ada-based SPARK verification toolset. The C toolchain will use the GCC compiler and CEA’s Frama-C platform. Both toolchains will be integrated within AdaCore’s IDEs.

Astrium Space Transportation will demonstrate the method and tools by deploying them on a major project to redevelop the software systems of its Automated Transfer Vehicle, aiming to prove the advantages of formal verification. Thales Communications will also use the project tools across its component-based middleware solution, adding the ability to automate the verification of generated code by using Hi-Lite annotation language.

By defining a common language of annotation for testing, static analysis and formal proofs, Hi-Lite will allow industries to switch gradually from an all testing policy to a faster and more cost-effective use of verification methods.

It loosely integrates formal proofs with testing and static analysis, thus allowing projects to combine different techniques around a common expression of properties and constraints.

The Hi-Lite project is primarily driven by the planned Formal Methods Technology Supplement of the DO-178C avionics standard. For the first time, this allows formal verification tools to replace physical testing when applying for system certification. As well as aerospace and defense, the products created through Hi-Lite aim to make formal verification available and easier to use across more industries, such as medical and automotive.

For more information and regular updates on the Hi-Lite project visit

http://www.open-do.org/projects/hi-lite

Quotes from the Hi-Lite project partners

"As high integrity systems get larger and more complex, formal methods provide a cost-effective solution that decreases the need for testing and speeds up project completion," said Arnaud Charlet, Hi-Lite Project Leader of AdaCore.

"Working with the Hi-Lite project partners, we aim to make formal verification faster and easier to use across large, multi-language projects that need to meet certification criteria, such as the forthcoming DO-178C standard."

"Altran Praxis are delighted to be involved in Hi-Lite. We look forward to the project bringing the advantages of formal verification and the SPARK approach to a wider audience in the software development community," said Keith Williams, Altran Praxis Managing Director.

"CEA-LIST will provide its expertise on the Frama-C platform for the formal proof of C-based software. We will work on connecting the Ada and C parts of the specifications and proofs," said Loïc Correnson, leader of the Frama-C team at CEA-LIST. "Indeed, as a co-author of the ACSL specification language, we will participate in the elaboration of the specification language(s) in the project."

"The ProVal Team at the INRIA Saclay research center is pleased to be a member of the Hi-Lite project," said Claude Marché, Senior Research Scientist at INRIA.

"We will provide expertise in automated reasoning and needed improvements in the Why/Alt-Ergo deductive verification tool-chain."

[…]

Indirect Information on Ada Usage

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

Job offer [United Kingdom]: Embedded Software Engineer

Embedded software development Engineer to perform integration and debugging of auto-coded C functional blocks into a hand-written Ada infrastructure, with particular emphasis on the optimisation of run-time performance on PowerPC-based and/or Intel Atom-based targets.

[…]

Analysis of the shortcomings in the current software architecture and design of improvements.

Identification of potential improvements at the Simulink level and liaison with the algorithm designers to get these implemented.

The successful candidate must be able to demonstrate a high degree of competence in at least one of the following areas:

- Matlab/Simulink
- Ada
- C
- VxWorks 6.4
- Virtex 5 FPGA technology
- PowerPC
- Intel Atom

[…]

Job offer [United Kingdom]: Software Verification and Validation Engineer

An exceptional opportunity for a talented Software Test Professional with experience of Verification and Validation processes within safety-critical engineering.

[…]

Software Verification and Validation Engineer key responsibilities:

- Prepare /update product tests specifications and product integration test specifications.
- Run and record product tests and integration tests using the factory test facilities.
- Prepare and run software module tests involving the use of software test tools.
- Carry out verification tasks on the product software development, and document the results in verification reports. Assist in the production of product validation reports.
- Assist with the reviews of software designs and other software documentation.

Software Verification and Validation Engineer Skills and Experience Required:

- Experience of software verification / testing, ideally within a safety-critical environment (aerospace, rail, defence, nuclear, medical devices etc….)
- Experience of software verification.
- Knowledge of programming languages C, C++, C#, Ada.

Job offer [United Kingdom]: Simulation and Modelling Engineer

Simulation and Modelling Engineers are required to fulfil existing and future contracts in the development of some of the worlds most advanced future defence systems. Our Simulation & Modelling Engineers have a proven track record in understanding and designing complex weapon systems to meet demanding customer needs. […]

Typical duties would include:

- Prepare a full range of specifications, develop designs in line with demanding customer specifications.
- Carry out a full range of technical analyses and investigations, including evaluation and selection of technical options.
- Carry out investigations into a full range of problems, issues or developments and develop and prepare solutions, individually or as a member of a project team.

Academic Qualifications

An appropriate Engineering Degree or Masters e.g. Systems or Aerospace Engineering, Electrical Engineering, Physics, or equivalent.

Experience Requirements

Proven track record in the following

- Requirements capture including interface definitions
- Simulation & Modelling
- Matlab and preferably also Simulink

Ideally will have knowledge and experience of the following:

- Doors or UML
- Ada and/or C

[…]

Job offer [United Kingdom]: ISVV Test Engineer

[…] a Test Engineer who will perform verification and validation testing on safety-critical components of a system to internationally recognised standards.

This task involves:

- Static analysis verification;
- Design and code scrutiny/verification;
- Using synthetic and simulation environments to design and test design models;

- Writing module test specifications, executing tests and analysing results;
- Performing independent assessment of other unit test results;
- Communicating results with a high degree of accuracy and attention to detail.

Outside this project, the company has a number of further challenging V&V activities as well as analysis, architecture, design and development projects that the successful candidate would also be expected to participate in.

You may be expected to contribute to bid work and internal research and development activities.

Essential Skills & Experience […]

- A proven track record in ISVV activities for safety-critical systems preferably for avionics components;
- A BSc or higher in a numerate degree (e.g. Computer Science);
- Experience in C, C++ or Ada;
- Model, design, code and static analysis verification;
- Unit test, strategy definition, test case definition and test execution;;
- Experience in using LDRA;
- Experience in using configuration management tools;
- Safety-critical related standards (e.g. DO-178B, EN 50128);
- System/software risk and issue identification and documentation;
- Familiarity with modelling languages and tools such as UML, SysML and Enterprise Architect;

[…]

Desirable Skills & Experience […]

- Experience in defence or avionics sectors;
- Definition of verification and validation strategies and plans;
- Experience writing macros and VBA process rationalisation formulas;
- Requirement capture (DOORS);
- Awareness of software Quality Management practices;
- Systems Engineering.

Job offer [United Kingdom]: Senior Software Engineer

[…] looking for a Senior SWE with expertise in VHDL, Embedded C, Ada, strong knowledge of hardware designs & principles, machine level programming, Assembler, this will be working on safety critical software, it is essential that you have a good degree. […]

Job offer [Spain]: Embedded software engineer

Looking for software engineers with experience in the development of real-time software (design, software architecture, coding, unit testing, integration testing, validation), with working knowledge of one or more programming languages (C, C++, Ada etc.).

[…]

Requirements: At least two years' experience as Embedded Software Engineer

Skills and experience:

Microcontrollers (Hitachi, ST, Intel, Motorola…), real-time operating systems (VxWorks, Nucleus, pSOS, MCarlo etc.), programming languages(C, C++, Ada, TCL, Visual Basic), communication buses (CAN, Most, I2C…), configuration management tools (Clearcase, Continuus…).

[translated from Spanish —mp]

Job offer [Spain]: Embedded software engineer

[…]

You will work in a project for the avionics domain as system integrator.

Required knowledge: Ada, Assembler, Perl, C, C++

[translated from Spanish —mp]

Job offer [United States]: Systems Engineer

This position is responsible for a variety of engineering assignments.

Employee understands and applies good system engineering concepts and practices, has a thorough understanding of software engineering development and verification concepts throughout the complete software lifecycle, and follows established policies and procedures to complete work assignments.

The successful candidate must meet the following basic requirements:

- BS in Software Engineering, Computer Science or related field.
- 5+ years of software engineering experience in real-time embedded systems using C, C++, or Ada
- 1 - 3 years of systems engineering experience in the avionics or aerospace industry.

The successful candidate will possess:

- Knowledge of C/C++, Ada, and Python or Perl
- Experience with Model-Based Development tools
- Strong inter-personal and communication skills
- Experience on software systems developed to DO-178B Level C or higher
- Understanding of software verification under DO-178B
- Experience with the development and formal verification of avionics software systems
- Ability to define and develop system requirements
- Experience with embedded real-time software development and verification
- Ability to perform analysis of requirements, design, development, verification, and documentation of software applications.
- Broad knowledge of avionic systems.
- Knowledge of aircraft data protocols such as ARINC 429, RS-485, RS-232

[…]

- Knowledge of the SEI CMMI processes and procedures
- Windows CE development experience is a plus
- OpenGL or GEODE experience is a plus

Ada in Context

GNU Go Ada Initiative

From: David Sauvage <contact@gnugoada.info>
Date: Tue, 9 Mar 2010
Subject: Preparing the GNU Go Ada Initiative
URL: http://blog.gnugoada.info/post/2010/03/09/first

Hello world,

we are a bunch of millions of developers and users.

We want our libre software we create, develop, maintain & use to exist as long as human beings exist, to preserve our citizen freedom and such other things …

We need the corresponding technologies to make it as efficient as it could be, because we often don't have infinite energy to get all of those done…

That's also why Ada has something to bring to the GNU Community.

Goals :

- Communicate & Promote to the GNU community about the amazing open Ada technology that is available.
- Present and promote all Ada Open Source Projects & Initiatives that can interest the GNU Community.
- Promote software engineering to the GNU Community

Get prepared :

- Brainstorm about the GGA Initiative, complete the goals
- What tools do we need ? (Wiki, …)
- How to communicate nicely ?

- Brainstorm about how to launch the GGA initiative (Contents, Events, Comic Strips, Tutorials …)

Act :

- Create categories for all kinds of subjects (Software Engineering, Process, Technology, Education, …)
- For each category create the key ideas we want to share
- For each category create the contents, events, comics, tutorials, …
- For each category add the links to other Ada websites, projects & initiatives
- …

To get redactor permission on the GGA Initiative, please subscribe at http://www.gandi.net/login/new and send your login name at [1], I will add it to the redactor list.

Do not hesitate to discuss about it on your preferred newsgroups, may be a redactor would input the discussion synthesis on the blog.

[...]

[1] contact@ this host name (without blog.)

On the TIOBE index and the ranking of Ada

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Mon, 8 Mar 2010 03:53:34 -0800 PST
Subject: TIOBE index
Newsgroups: comp.lang.ada

Hello,

From time to time I'm looking at the TIOBE Programming Community Index

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

The results, for long, are not very glorious for Ada, but I told myself if was like it was.

For instance the latest ranking for Pascal, Fortran and Ada are:

Tiobe ranking

Pascal 0.603%

Fortran 0.563%

Ada 0.376%

Now, I spent a few minutes to look at the figures myself, following the famous citation "The only statistics you can trust are those you falsified yourself".

So I made the search engine queries as described there:

http://www.tiobe.com/index.php/content/paperinfo/tpci/tpci_definition.htm

including 1-year time windowing when available.

Here are the results (the simple queries without time windowing give similar results):

	Google	Google Blogs	MSN
Pascal	2'020'000	25'027	3'170'000
Fortran	1'510'000	11'768	1'640'000
Ada	2'140'000	110'310	2'670'000

	Yahoo!	Wikipedia	YouTube
Pascal	25'200'000	1'664	232
Fortran	7'900'000	949	113
Ada	49'300'000	1'764	70

With the weights being 23% for each engine, except YouTube with 7%, it is obviously impossible to have Ada's ranking that is lower than Fortran's and even than Pascal's.

Funny thing: the weights sum up to more than 100%, so don't forget to divide by the sum of weights!

If you take for instance Pascal's ranking as a reference, the rankings would be actually the following:

Pascal 0.603%

Fortran 0.294%

Ada 0.668% <- uh-oh, almost the "A" category...

Of course the reference (here Pascal) would need as well a re-ranking, up or down, which would influence Fortran's and Ada's - this would mean to gather the full index data!

Did I miss something ? Some "soft factors" ?

I have an Excel sheet for those interested.

Or should we bug TIOBE about their statistics' quality ?

Or publish a fair, transparent index based on the TIOBE idea, but with the publication of the figures ?

From: jonathan <johnscpg@googlemail.com>
Date: Mon, 8 Mar 2010 14:04:46 -0800 PST
Subject: Re: TIOBE index
Newsgroups: comp.lang.ada

[…]

Thanks Gautier … I hadn't realized how silly the TIOBE index is until you explained how it works.

Another data point:

http://www.blackducksoftware.com/oss/projects#languageos

Still can't draw any real conclusions about the software industry from this data point, but at least it is not meaningless.

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Mon, 8 Mar 2010 21:39:04 +0200
Subject: Re: TIOBE index
Newsgroups: comp.lang.ada

[…]

There is also http://langpop.com/ which I find slightly more realistic based on my own experiences, although that doesn't show Ada in very popular light either.

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Mon, 8 Mar 2010 12:22:32 -0800 PST
Subject: Re: TIOBE index
Newsgroups: comp.lang.ada

[…]

That would not be a big issue for me - provided the conclusions can be reproduced.

I only have something against dubious statistics…

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Mon, 08 Mar 2010 16:41:22 +0100
Subject: Re: TIOBE index
Newsgroups: comp.lang.ada

> Did I miss something ? Some "soft factors" ?
>
> I have an Excel sheet for those interested.
>
> Or should we bug TIOBE about their statistics' quality ?
>
> Or publish a fair, transparent index based on the TIOBE idea, but with the publication of the figures ?

Well, ask them - they usually answer.

Apart from that they lost my personal credibility whey they dropped usenet if favour of hipper stuff like YouTube. And when all first 20 languages where in "danger" to be "A" rank they changed something else. I wrote a blog entry about that:

http://ada-programming.blogspot.com/2008/01/tiobe-january-2008.html.

Also I noticed: "The first 100 pages per search engine are checked for possible false positives and this is used to define the confidence factor" - the confidence factor for Ada is not publicised.

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Mon, 8 Mar 2010 08:26:31 -0800 PST
Subject: Re: TIOBE index
Newsgroups: comp.lang.ada

Well, newsgroups contents are visible, copied several times, on various search engines…

YouTube has a normalized weight of only 5.73%.

> Also I noticed: "The first 100 pages per search engine are checked for possible false positives and this is used to define the confidence factor" - the confidence factor for Ada is not publicised.

That could be the explanation…

Anyway, the idea of a "transparent" index, with the same rules, but with publication of all figures, could be a nice project.

Of course automatized…

[…]

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Tue, 9 Mar 2010 12:48:31 -0800 PST
Subject: Re: TIOBE index
Newsgroups: comp.lang.ada

OK, first embryo here:

http://sf.net/projects/lang-index/

Already working for some engines :-)

[…]

Update on the AuroraUX project

From: AuroraUX Website
Date: Sun, 4 April 2010 [last update]
Subject: AuroraUX Project Goals
URL: http://confluence.auroraux.org/confluence/display/AUXREQ/AuroraUX+Project+Goals

[…]

The AuroraUX Operating System is uniquely designed to provide a high-integrity computing environment for high-end desktop and workstation users which require a rock solid, fast computing platform with no compromises.

AuroraUX shall endeavor to provide a secure high integrity operating platform to build mission-critical computing environments for desktop workstations that simply cannot waste time. Our approach to this is by carefully thought out designs for our system libraries and accompanying applications that target key areas of modern desktop computing that seem to fail frequently.

By use of the Ada language, a proven and tested high integrity software development model normally used in the aviation industry as well as certain code standard […] and peer review, we aim to provide the most stable desktop operating system available on the market today.

Summary:

- A Unix Shell Interpreter that conforms to the POSIX spec much like ksh93 written in Ada.
- Simple, clean boot loader.
- Port the GNAT Ada compiler direct on top of LLVM.
- Entire tool-chain based around LLVM and not GCC !
- Support for Ada, C, C++, D as well as Fortran.
- A graphical session manager written in Ada to replace convoluted solutions such as GDM, KDM etc…
- A modern window management framework written for the twenty first century where highly interactive, asynchronous, fault tolerant, predictable behavior is paramount.
- Provide a modern OpenGL type library toolkit for Ada programmers, such like the Clutter toolkit as part of the system core libraries.
- Long term goal of replacing the 'X server backend' and mesa userland driver, kernel drivers and all that mess with a in-kernel, fault tolerant, modern asynchronous, object-oriented graphics subsystem.
- A core system with a highly advanced, flexible and stable packaging system with built-in system level package integrity checking to ensure reliable system updates and the ability to roll back automatically.
- Enable the user to run popular programs as found in other *nix's, such as GNU/Linux 2.6 via ABI system call emulation.
- Excellent and Accessible documentation.

[…]

[To check how project goals have evolved in the last 12 months, see "AuroraUX project" in AUJ 30-2 (Jun 2009), p.84 —mp]

GNAT AUX (DRACO) for AuroraUX

From: AuroraUX Website
Date: Fri, 16 April 2010 [last update]
Subject: GNAT AUX (Later DRACO) Milestones And Goals
URL: http://confluence.auroraux.org/confluence/display/AUXBP/GNAT+AUX+(Later+DRACO)+Milestones+And+Goals

Several people, including myself, have direct experience with AdaCore admittedly ignoring any platform in which they have no current paying customer.

This definitely includes Dragonfly as they refused my patches, and FreeBSD.

They would prefer patches get sent to the FSF, but they won't accept them without copyright assignment which apparently takes months to achieve, if you achieve at all. And that might be per-submission. The whole situation is just wrong.

On top of that, AdaCore won't provide a compiler that produces GPL-free executables. GCC does, but that only gets updated once per year, and it's still to be determined if the FSF version has all the features the GPL version has at the point of release. It's time to fork. At the very least, we get a version that anybody can easily contribute to, with all the current patches, and we can produce current versions on a much more frequent basis.

Milestones

Stage 1: Fork

At a strategic points, probably when GPL 2010 is released, we dump a subset of the GCC codebase into a Git repository (head revision). This would be gcc/ada + subdirs, /gnattools, gcc/testsuite/ada, and selected files from /gcc, /gcc/config + subdirs. Unlike the GCC, we don't maintain separate branches of Ada. Theoretically we just take a copy of GCC core and overlay the current version of our repository over it, and we should be able to build the latest possible Ada compiler. We add in all the patches that we want, and particularly cater for the BSDs (Open, Net, Dragonfly, and Free). From this point on, we monitor commits in the GCC head that affect Ada and make sure to mirror them in our own repository. The forks may diverge in time, but for a while they will be parallel and more or less compatible. It will be known as "GNAT AUX" until Stage 3.

Stage 2: DragonEgg

This will be a quick stage. We compile DragonEgg and add it as a plugin to our forked compiler. This will replace the GCC optimizers with the LLVM optimizers. We verify that it can build itself, and pass all the ACATS and GNAT testsuite, at least to same level as the GCC did.

Update: In its current state, DragonEgg can build GNAT, but then GNAT can't build it's gnatlib due to missing error handling / exception functionality. We are waiting for this functionality to be incorporated by Duncan Sands (baldrick) before continuing. [13 April 2010]

Stage 3: Streamlining

We'll branch the repository. The GNAT AUX branch will continue to be maintained and to use DragonEgg. The master trunk will take on the name DRACO.

[…] Upon further study, I'm rescinding the last proposal to incorporate DragonEgg into DRACO. While Edward in his comments treats the replacement of GiGi as trivial, ultimately it is the right approach and dealing with the complexity of building DragonEgg into DRACO is going to be a lot of wasted effort.

Stage 3 is still streamlining, here are the tasks as I see them now:

1. Create a new directory along side GNAT AUX trunk rather than a true git branch. This is because I'm going to organize the directory structure and remove all extraneous gcc code.
2. The new top-level directories of the branch will be: /driver, /frontend, /gcc-interface, /gcc-remnants, /llvm-interface

3. Objective 1: Replace the GCC driver with a brand-new one written in Ada. Basically this just accepts the switches and passes them to the front end. The GCC driver is designed to handle many different language plugins and DRACO will by design only know Ada. By replacing the C-based driver, we can eliminate a lot of C-code which becomes cruft when the compiler only needs to know one language. The code goes in the /driver directory. The Driver will be BSD licensed.
4. Objective 2: Replace the horrid GNU autotools configure and makefile stuff with a much more sensible CMAKE version. For starters, I want to reintroduce the explicit build stages (e.g. build dracolib, build dracotools, build cross-lib, build cross-tools, build cross-all, etc) rather than badly guess what the builder really wants. Secondly, the makefile will be monolithic rather than spread out over a dozen places. Obviously it will build what is exactly needed by DRACO at the specific development stage and nothing extra, so builds will be much faster. This code stays in the branch directory. By default, the CMAKE script will create a subdirectory called "build" so avoid polluting the source code during bootstrapping stages. GCC requires this, but forces you to set it up manually. CMAKE will use Clang to compile the C parts of Draco.
5. Objective 3: Eliminate as much GCC code as possible while maintaining a functional GiGi interface and GCC optimizers and code generators. This includes eliminating the C language, and as many libraries as possible. I definitely hope to eliminate mpfr, mpc, and gmp libraries as build requirements to DRACO at this stage. If some libraries need to stay a while longer (e.g. libgcc) then we limit the included functions to the ones needed by DRACO to function, nothing extra. All these "GCC remnants" go in the gcc-remnants folder with the goal to reduce this over time.
6. Objective 4: To complete the reorganization of the location of the code, the files of trunk/gcc/ada will move to draco/frontend and the files of trunk/gcc/ada/gcc-interface will move to draco/gcc-interface. For now, the llvm-interface folder remains a placeholder. Obviously the goal will be to delete the /gcc-interface over time.

Upon completion, Draco will not have any functional advantage over the latest GCC, and in fact has lost the ability to compile C and therefore will not have the ability to fully host itself as GCC can. This configuration doesn't require DragonEgg, so this stage can occur in parallel with Stage 2. The main benefits are a more sane build process, much less code to build (and therefore less opportunity to break during build), smaller executables, and a more clearly modular layout to replacement of GiGi and the GCC middle- and back ends.

Stage 4: Native Intermediate Representation (IR)

Even though we ripped out the GCC backend, we still need this 23MB compressed tarball worth of source files, just for GCC "Core".

We want to get rid of GCC completely. One way may be to use common parts of Clang like the parser and lexicon pattern recognition. Basically you would take clang, rip out the c-language parts and replace it with the Ada parts. I also think that the Ada compiler should not build any other language beside Ada, so at this point we would need Clang + an Ada compiler to build our new compiler. Alternatively we could just add Ada to Clang and possibly rip out C++ and Objective C. If we took this approach, the resultant could build itself. This might be the most prudent way to go, but it wouldn't be the minimalist approach.

DragonEgg is basically a stopgap measure. At the completion of Stage 3, DRACO is converting the Ada Abstract System Tree into the GIMPLE language which DragonEgg intercepts and painstaking converts into LLVM's IR language. This is inefficient in terms of time and optimization. We need to replace the Ada-to-GIMPLE translation (Ada-to-C) with Ada-to-IR (Ada to Ada with Ada Bindings to LLVM). The goal should be to eliminate all C files between the AST tree and the LLVM library. Once this is complete, DragonEgg's purpose is complete and it is no longer required. Clang and LLVM are the only components required to build a new DRACO, besides an existing DRACO.

The objectives of this stage are:

1. Primary Objective: create the "DLC" (Draco AST /LLVM Intermediate Representation Converter), which is the interface between Draco and LLVM.

It converts the native "expanded and decorated abstract syntax tree" into LLVM's language-independent Intermediate Representation.

The DLC will be written in Ada, unlike the 21K+ lines of C code that GiGi is written in. The DLC will be BSD licensed.

2. Objective 2: Replace GiGi with DLC. Delete all GiGi related code from /gcc-interface directory, and the directory itself if possible.
3. Objective 3: Delete all GiGi related code from the /gcc-remnants directory

Stage 5: Hardcore and Unnecessary

If we complete Stage 4, we pretty much should be popping champagne. We managed to completely separate the Ada compiler from GCC, something that it was built on. It would be like separating conjoined twins! To be purist though, there shouldn't be an "C" files in the compiler, it should be 100% Ada. If we wanted extra credit, we would port any non-Ada source files into Ada and bootstrap. If we take a significant port of Clang, this could be a large amount of work. It would fall under the "Because we can" category.

By now we've eliminated tens of thousands of lines of C since the streamlining purge. There is still some left though, like POSIX interfaces to threading and errno, and other bindings. Some of the C is in the /front-end folder and the the rest of it will be in /gcc-remnants assuming the /gcc-interface directory has already been purged from the DRACO branch. The ultimate goal should be to produce an Ada compiler than can fully build itself without another compiler (i.o.w. without Clang).

[…]

On Min/Max attributes and the type/subtype relationship

From: Alex Mentis <asmentis@gmail.com>
Date: Tue, 27 Apr 2010 12:34:18 -0700 PDT
Subject: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

I'm disappointed with some allowed syntax that seems a little error-prone.

Consider the following code:

```
with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;
procedure Main is
  Nat : constant Natural := 0;
  Pos : Positive;
begin
  Get (Pos);
  Put (Positive'Min(Nat, Pos)); -- Ada
  -- does not require the Min
  -- attribute to enforce a Positive result
end Main;
```

This program happily outputs that the minimum of (0 and whatever positive value you enter) is 0. Now, I concede that the program is working exactly as the ARM specifies. The Min (and Max) attribute functions accept and return types of S'Base, in this case Positive'Base. But doesn't it seem like a bit of a tease to allow a programmer to specify S'Min if the compiler is allowed to ignore the type of S in the function's parameter list and the program does not raise a Constraint_Error at run-time if it returns a value outside the range of type S?

If it's too hard to enforce strictly then maybe the functions should be named Unchecked_Min/Unchecked_Max. Or maybe the programmer should be constrained to using the attributes with only a base type. Or, at the very least, can't the compiler generate a warning about this? I turned on all warnings in GPS and got nothing.

Things that make you go hmmm…

[…]

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Tue, 27 Apr 2010 13:20:41 -0700 PDT
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

If you want the check, this should do:

```
begin
  Get (Pos);
  Put (Positive (Positive'Min(Nat, Pos)));
end;
```

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Tue, 27 Apr 2010 17:16:08 -0400
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

> Put (Positive (Positive'Min(Nat, Pos)));

That works, but I think a qualified expression is better:

```
  Put (Positive'(Positive'Min(Nat, Pos)));
```

By the way, there are lots of attributes that work like this (use the base subtype). It's not just Min and Max.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 27 Apr 2010 17:46:29 -0500
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

Right. Attributes are mostly type-related, not subtype-related. The name in the prefix serves to identify the type, not the subtype (there are no named types in Ada, only named subtypes).

All of 'Succ, 'Pred, 'Val, and 'Value work this way, and there are probably many, many more. Even 'First and 'Last technically work this way (although it doesn't matter in that case).

From: Alex Mentis <asmentis@gmail.com>
Date: Wed, 28 Apr 2010 03:36:40 -0700 PDT
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

I think you all missed my point: allowing the programmer to specify a constraint in situations where the constraint will not be enforced could cause confusion/error for someone who doesn't have the entire ARM memorized…

Yes, I know there are ways to force Ada to do the check, like explicit type conversion and qualified expressions. That's pretty ugly, though. If I already have to put the type in front of the attribute, why should I have to put the type in front of that again? As long as we're posting work-arounds, though, I suppose I would use the following:

```
with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;
procedure Main is
  function Min(Value_1,
               Value_2: Integer)
    return Integer
    renames Integer'Min;
  Nat : constant Natural := 0;
  Pos : Positive;
begin
  Get (Pos);
  Put (Positive'(Min(Nat, Pos))); -- return
  -- a Positive result from renamed Min
  -- else raise a Constraint_Error
end Main;
```

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Wed, 28 Apr 2010 03:58:47 -0700 PDT
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

> I think you all missed my point: allowing the programmer to specify a constraint in situations where the constraint will not be enforced

But the prefix of an attribute reference does not specify a constraint.

Integer'Min, Positive'Min, Natural'Min are all the same.

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Wed, 28 Apr 2010 04:37:35 -0700 PDT
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

But it should, at least in the case of Min/Max.

Integer'Min, Positive'Min, Natural'Min being all the same breaches the spirit of the Ada language (what you see is what it means).

A topic for Ada 201z…

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Wed, 28 Apr 2010 04:47:42 -0700 PDT
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

We could have had is thus and unconstrained Min as Positive'Base'Min when Min was introduced…

But it is as it is, and your proposal would be a severe incompatibility, so it won't fly.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 28 Apr 2010 15:41:16 +0200
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

Well, the proposal might be to fix rather the issue of the superfluous subtype specification. Obviously Max (and many other attributes) are primitive operations and need no subtype to specify. So:

```
  X'Succ, X'Pred, X'Image
```

Specifically max and min should be a dyadic operations:

```
  function "max" (Left, Right : T)
    return T'Base;
```

(and, please, no new reserved keywords!)

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Wed, 28 Apr 2010 16:10:46 +0200
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

That'll be fun:

```
C'Succ'Succ
'C'&'&"Succ
""'Succ
(M + N)'Succ
```

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 28 Apr 2010 16:53:25 +0200
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

This is a slightly different design flaw. Things like "abc"'Length, "abc"'First are illegal in Ada.

No fun! BTW, if you prefer dotted notation it could be

```
X.Succ
```

as well.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 28 Apr 2010 16:07:24 -0500
Newsgroups: comp.lang.ada
Subject: Re: Min/Max attribute makes promises it can't keep

[…]

That's not a real problem. Ada 2012 will allow

```
String'("abc")'Length
```

As a qualified expression can be used as a name (it's considered constant).

If you think this is weird, Tucker points out that you can already write:

String(String'("abc"))'Length

in Ada 95, as a type conversion is a name. We did wonder if any compilers could actually handle it, but given that this long-winded locution is already legal, the shorter one might be legal as well.

As for the initial concern about giving a subtype name, in the case of literals you have to give one somewhere (since a literal can be of many different types, and the results can vary depending on the type used -- not for 'Length, but for 'Last and most other properties).

We did talk a bit about object attributes like suggested back a few messages for Ada 2012. We didn't get much consensus, mainly because I think people were looking at the wrong questions.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 29 Apr 2010 00:17:05 +0200
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

No problem, Ada supports overloading in the result type. E.g. abs (-1) is OK.

(I have an impression that many weird Ada rules rooted in an attempt to express them at the grammar level.)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 1 May 2010 00:42:53 -0500
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

That doesn't help. Consider:

```
type My_String is
  array (Natural range <>) of Character;
```

Now, if you wrote:

```
"ABC"'First
```

The answer depends on whether the type of the prefix is String or My_String, but the type is Integer'Base in both cases. No amount of overloading will help.

The language does not want the compiler to have to list out all of the possible types for a string literal, figure out the appropriate bounds for all of them, take all of the ones with the appropriate index type, and then allow the attribute if the answer is the same. (Recall that *every* Ada program has at least three string types available.)

Similar rules are used for aggregates, the operand of a type conversion, and other places. No context can be used as figuring out an answer if it could is just too difficult. (There are a couple of exceptions for attribute prefixes, particularly 'Access, as we found that the typical rule is too limiting, but those are tightly bounded.)

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 1 May 2010 08:28:10 +0200
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

[…]

How is that different to:

```
package P is
  function ABC return String;
end P;
package Q is
  type My_String is array
     (Natural range <>) of Character;
  function ABC return My_String;
end Q;
use P,Q;
…
ABC'First; -- Ambiguous
```

> The language does not want the compiler to have to list out all of the possible types for a string literal, figure out the appropriate bounds for all of them, take all of the ones with the appropriate index type, and then allow the attribute if the answer is the same.

That would be silly, of course. But it is not what I meant. Attribute should be an overloaded operation as any other. If the expression happens to be ambiguous there are language means to specify what it is.

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Wed, 28 Apr 2010 21:41:57 -0700 PDT
Subject: Re: Min/Max attribute makes promises it can't keep
Newsgroups: comp.lang.ada

> That's not a real problem. Ada 2012 will allow
>
> String'("abc")'Length
>
> As a qualified expression can be used as a name (it's considered constant)

This whole discussion begs the question "What is an attribute?". I think the Ada 83 design idea was that the basic scalar types only have operators as functions. (See package Standard, there you find only operators (except for the fact that enumeration literals are also functions)). Everything else (like 'Image) was defined as an attribute.

Thus Min, if it had existed in Ada 83 as an operator (it hadn't even as an attribute), would have had to be a reserved word like abs, not, rem, mod.

I think Ada 95 kept this design issue and thus had to define it as an attribute.

On Stream_Access and Ada.Streams

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Fri, 07 May 2010 03:02:16 +0200
Subject: Why is not Stream_Access defined Ada.Streams ?
Newsgroups: comp.lang.ada

[…]

My dirty stuff of the day : why is Stream_Access not defined once a time in Ada.Streams and is instead defined elsewhere multiple times, like in Ada.Streams.Stream_IO, Ada.Text_IO.Text_Streams, Ada.Wide_Text_IO.Text_Streams and Ada.Wide_Wide_Text_IO.Text_Streams?

Isn't it funny design ?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 6 May 2010 21:24:49 -0500
Subject: Re: Why is not Stream_Access defined Ada.Streams ?
Newsgroups: comp.lang.ada

[…]

Ada 95 didn't have anonymous access returns, so a named type had to be defined. I suspect that there was no intent that it be used for anything other than defining the result of these functions -- and there was no intent that that result be stored: just directly dereferenced and passed to a stream attribute.

If we were writing it today, I'm pretty sure that no named type would be used at all. But making a change like that now would be incompatible.

On the efficiency of child and nested packages

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Sat, 29 May 2010 14:56:53 +0200
Subject: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

For some reasons, I've turned some nested package -- packages nested in the body of package -- into a set of child packages instead.

I've noticed the application's execution time is now an average of 125% of that of the old implementation.

The implementation did not changed otherwise and is still the same.

What can make invocation of subprograms slower when the subprograms are in a child package rather than in a nested package?

Note: this is with optimization enabled -- with -O -- and both old and new implementation was compiled with the same options.

I can't explain that and can't imagine a reason why. That's a mystery to me.

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 1 Jun 2010 11:35:02 -0700 PDT
Subject: Re: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

Something else that occurred to me: In the process of turning the nested packages into child packages, were there any global variables declared in the body of the parent package that you had to move to the spec (probably the private part) so that they would be visible to the child packages?

From: BrianG <briang000@gmail.com>
Date: Sat, 29 May 2010 21:17:51 -0400
Subject: Re: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

Presuming you're using GNAT (a guess based on your previous posts), could using -gnatN (as opposed to -gnatn) help?

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Tue, 01 Jun 2010 21:09:37 +0200
Subject: Re: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

No Adam, there was not. There were global variables, but just globals in child packages, not in parent package.

I've just found : pragma Inline is not automatically applied, it requires the

-gnatn option Brian talked about. I though this was automatically done when the -O option is present (I rarely use inline, so I was not aware of that).

Anyway, I'm interested in your experience: what did occurred to you with global variables ?

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 1 Jun 2010 12:44:46 -0700 PDT
Subject: Re: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

It was just a theory. My thinking was that if you have a global variable in a package *body* (and no subunits), a compiler can, in theory, draw some conclusions about how the variable is used, since there can be no uses of the variable except by subprograms in the package body, and perhaps perform some optimizations based on that.

This doesn't work if the variable is in the spec (even in the private part), since the compiler won't know what child packages might be added later that have access to the variable. Anyway, this was just a theory, not really based on any experience.

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 1 Jun 2010 08:03:44 -0700 PDT
Subject: Re: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

I can't say anything about any particular compiler. One possibility: if procedure A calls procedure B, and A and B are in the same source, a compiler may be able to put the code of B inline in procedure A's code---i.e. the code for A will include B's code, rather than including a "call" instruction. If it does this, then in the process, it may also to be able to eliminate instructions in B's code that have no effect on A. It's a lot harder to do this if A and B are in different sources, which I'm assuming is happening if you are pulling code out of a nested package and putting it in a child package.

Again, this is just a wild guess; without knowing anything about your source, and with my limited knowledge of GNAT, I can't say anything for certain.

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Date: Tue, 01 Jun 2010 17:34:22 +0200
Subject: Re: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

Adam, I've thought about it at that moment, so I've added some pragma Inline for the relevant subprograms in the packages specs. This did not change anything.

Do you think there are some reasons to believe pragma Inline is not properly applied with inter-package subprograms invocations ?

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 1 Jun 2010 08:38:54 -0700 PDT
Subject: Re: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

> Adam, I've thought about it at that moment, so I've added some pragma Inline for the relevant subprograms in the packages specs. This did not change anything.
>
> Do you think there are some reasons to believe pragma Inline is not properly applied with inter-package subprograms invocations ?

I can't say. Different compilers will handle this differently. The language standard says that Inline pragmas don't have to be obeyed---they are just suggestions.

The only real way to tell what's going on is to disassemble the code.

From: Simon Wright <simon@pushface.org>
Date: Tue, 01 Jun 2010 20:04:44 +0100
Subject: Re: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

> Do you think there are some reasons to believe pragma Inline is not properly applied with inter-package subprograms invocations ?

First, you may need to use -gnatn or -gnatN and at least -O2 (have to look up the manual for this).

You could also try the GNAT special pragma Inline_Always.

I've found (powerpc-wrs-vxworks) that inlining can in fact slow things down. Cache effects, I suppose.

From: Alex R. Mosteo <alejandro@mosteo.com>
Date: Wed, 02 Jun 2010 13:11:16 +0200
Subject: -gnatN breakage was: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

I've played with -gnatn and -gnatN in the past, and found that the latter tends to bomb the compiler quite a lot. (GPL2008 and 2009 experiences).

Anyone seeing the same?

From: Vadim Godunko <vgodunko@gmail.com>
Date: Wed, 2 Jun 2010 07:39:49 -0700 PDT
Subject: Re: -gnatN breakage was: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

-gnatN is obsolete, you must use pragma Inline/Inline_Always with -gnatn instead.

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Wed, 02 Jun 2010 18:45:09 +0200
Subject: Re: -gnatN breakage was: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

Does this mean that 3rd party source cannot be inlined across units unless GNAT specific pragma Inline_Always is added to these 3rd party sources?

From: Vadim Godunko <vgodunko@gmail.com>
Date: Wed, 2 Jun 2010 13:35:29 -0700 PDT
Subject: Re: -gnatN breakage was: Child vs nested package : efficiency matter
Newsgroups: comp.lang.ada

[…]

No. Use of -gnatn and -O is sufficient to enable inlining of subprograms marked by pragma Inline. But not all subprograms will be inlined but only those definitely simple.

[…]

Parallel processing and latency in multi-core processors

From: Anatoly Chernyshev <achernyshev@gmail.com>
Date: Tue, 9 Mar 2010 09:48:37 -0800 PST
Subject: Multicore problem
Newsgroups: comp.lang.ada

[…]

Here is the following problem: large 3D array, which can be processed in parallel. Naturally, I was trying to make use of 2 cores on my processor by splitting the whole calculation into several parallel tasks. Strangely, I could actually see that the program utilizes both cores now (even not by 100% as I wished), but the computation time increases more than 2 times when there are 2 tasks running on 2 cores. During calculation any 2 or more tasks can occasionally modify the same array member, for which a protected type is introduced.

Here are results of benchmarking for my 2 core system (Intel P4 Dual Core, Win XP SP3):

1 task (50% total proc. load): 680 sec.

2 tasks (70-93% load): 1520 sec.

4 tasks (70-93% load): 1195 sec.

Any ideas on why it might happen and how to get around it?

Thank you.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Tue, 09 Mar 2010 20:10:02 +0200
Subject: Re: Multicore problem
Newsgroups: comp.lang.ada

[…]

Perhaps the two tasks/cores are competing for data-cache space, and causing more cache misses than if a single task/core is active?

How large is your 3D array, compared to the cache?

If this is the problem, you could try to divide the whole problem into slices so that each slice processes a part of the 3D array that *can* fit in the cache, but can also be processed in parallel by two tasks/cores. Not knowing what your processing actually does I can't say more about how to slice the problem.

From: Anatoly Chernyshev <achernyshev@gmail.com>
Date: Tue, 9 Mar 2010 10:48:34 -0800 PST
Subject: Re: Multicore problem
Newsgroups: comp.lang.ada

> Perhaps the two tasks/cores are competing for data-cache space, and causing more cache misses than if a single task/core is active?
>
> How large is your 3D array, compared to the cache?

Right now it is small, 23 MB, for the purposes of algorithm honing.

2MB L2 cache. In perspective the array could be 4-8 GB (not for my current machine, of course).

> If this is the problem, you could try to divide the whole problem into slices so that each slice processes a part of the 3D array that *can* fit in the cache, but can also be processed in parallel by two tasks/cores. Not knowing what your processing actually does I can't say more about how to slice the problem.

Each task begins in the cell of its designated area of array, and then does a Markov walk all around the place using the cell pointers.

From: jonathan <johnscpg@googlemail.com>
Date: Tue, 9 Mar 2010 16:08:26 -0800 PST
Subject: Re: Multicore problem
Newsgroups: comp.lang.ada

[…]

Of course I can't say much about your specific problem. I can tell you (the little) I learned battling something similar (using linux and GNAT). The problem I was up against was latency: minimum time to send data, no matter how small the data, between 2 cores on the same processor. I measured this time as about 40 microseconds, (exchange of data between cores). Might have been 20 or 60, but think 10's of microseconds. If message passing (or sharing data in an array) is more frequent than this, you'll never see a speed up on the several cores. (Some expensive networks advertise latencies of a few microseconds, and some experimental operating systems advertise sub-microsecond latencies.) So you should not think of passing data more frequently than say every 200 or 400 microseconds if you want to see speedup. Which is a long time .. you can do 10's of thousands of floating point operations in that time!

From: Karl Nyberg <karl@grebyn.com>
Date: Tue, 9 Mar 2010 14:37:02 -0800 PST
Subject: Re: Multicore problem
Newsgroups: comp.lang.ada

[…]

We recently had a thread on this kind of stuff on LinkedIn.Things like cache size and communications overhead (tip of the hat to Fred Brooks and the Mythical Man-Month here) all come into play.

LinkedIn Groups

Group: Multicore & Parallel Computing

Subject: New comment (7) on "Why does speedup decrease with the number of threads?"

In a system such as an Intel X86, you can execute several instructions per clock cycle. However, it takes several cycles to read or write data to L1 cache, more cycles to read/write to L2 cache, and even more cycles to read/write to L3 cache. If you have to read from memory, you might have to wait hundreds if not thousands of cycles for the read to occur. While the CPU is waiting for the read to complete from L1/L2/L3/ main memory, the CPU cannot execute. Also, each memory reference must have a valid TLB (translation lookaside buffer) to translate the virtual address to a physical address. I don't know what Windows 7 uses for a page size, but most operating systems since the 1970's use a 4k page size, and you might have 64 TLBs per core. You need to execute a system routine every time you get a TLB miss, and there are 100's of instructions in the TLB miss handler, as well as the supervisor context switch.

Think of memory like a disk drive. With DRAM, first you have to seek to the page in the memory device, then you need to access the block that you want, then you transfer that block into the CPU cache. There is a memory controller that serializes requests to memory between the CPU and external devices such as the GPU, disk, etc.

Now what happens when you go from 1 to 2 to 4 to 8 cores? Well, instead of 1 CPU making requests on memory, you have 2 or 4 or 8 CPUs making requests on memory. The memory controller needs to queue these requests and operate on them one at a time. So when you go from 2 cores to 4 cores, you can double the size of the queue waiting for a memory read to occur. If 1 read takes 1000 cycles, and you now have a queue 4 deep, that 1 read can now wait for 4000 cycles worst case, or 2000 on average. With 8 cores, you can now have a queue 8 deep when each core issues a memory read request. Don't forget that your graphics card will be DMAing to and from DRAM as well, so this can cause memory delays, and you may have other I/O devices, and other processes running in your system such as virus scanner, email, …

You can also have issues if you do not align the data between cores on cache line boundaries. If 2 cores try and access data in the same cache line, their respective caches can fight over the data when there is a write request, as only 1 core can 'own' a cache line, and it is expensive to transfer a cache line between cores.

Also, when you use 1 core, it has full access to the shared L2 and shared L3 cache. I have seen in Intel architecture where cores 0 and 1 share an L2 cache, and cores 2 and 3 share another L2 cache. So in fact, you may get better results using cores 0 and 2 when running 2 cores, instead of using cores 0 and 1. When you just use core 0, it has use of the full shared L2 cache. When you start using core 1, this core will also start using the L2 cache that is shared with core 0.

This means that data that core 0 needs will be evicted by core 1 and vice versa, thus causing both cores to slow down as the cores stall waiting for the cache to be refilled (thrashing).

Note that every time the operating system does a task switch, the contents of the TLBs must be flushed from the old task context, and regenerated for the new task. This is an expensive operation if you are not doing much computation in your task. Also, the cache may need to be refilled based on the new task address space.

When you go to use multiple cores, you should lock a specific task to a specific core, as it is expensive for a process to move between cores.

Also, when doing this sort of computation, you should be making use of the SIMD instruction set, where you can operate on 4 or 8 or more data items in 1 instruction per core.

Hyperthreading just means you have virtual cores. It is a mechanism to keep the physical core busy executing instructions while a L1/L2/L3/ memory transfer occurs. If you program correctly, you should be able to fully utilize the power of the virtual cores.

Posted by Paul Burega

On default discriminants and mutable objects

From: Peter C. Chapin <pcc482719@gmail.com>
Date: Wed, 05 May 2010 20:41:57 -0400
Subject: Question about default discriminants and mutable objects.
Newsgroups: comp.lang.ada

Hopefully I can ask this question clearly…

I understand that an instance of a type with default discriminants can be mutated (here I mean have its discriminant changed) via assignment. It seems like the most effective way for a compiler to support that ability is to always allocate enough memory for the object to account for all possible discriminant values. Mordechai Ben-Ari pretty much says this explicitly in his book "Ada for Software Engineers."

I also understand that an instance of a type without default discriminants can't be mutated in this way (that is, by assignment). If a new value is assigned to the object with the wrong discriminant the result is Constraint_Error. This would allow the compiler to allocate just the memory necessary for that particular discriminant value used to initialize the object since there would never be a (successful) attempt to stuff a larger object into that space.

It seems like conceptually the issue of default discriminants and mutability (in the sense I mean here) are independent. One could imagine some currently non-existent syntax that would allow the programmer to mark a type declaration so that the compiler allowed discriminant values to be changed via assignment without leaning on the mechanism of default discriminants.

Furthermore one could imagine treating default discriminants as 100% syntactic sugar and not endowing them with any special semantics regarding mutability.

Okay, so my question is: what was the rationale behind combining the notion of default discriminants with the notion of mutability? Is there some subtle technical reason why they must go together? I don't have pressing need to know… I'm curious and I'm trying to deepen my understanding of this topic.

[…]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 5 May 2010 20:26:48 -0500
Subject: Re: Question about default discriminants and mutable objects.
Newsgroups: comp.lang.ada

[…]

> Okay, so my question is: what was the rationale behind combining the notion of default discriminants with the notion of mutability? […]

I don't know; I know the idea come from Ada 83, but I have no idea why they chose that.

It's been a long-time annoyance in Ada that these two concepts are intertwined. (It's very much like the annoying requirement that a protected function do only read-only things to the protected object, while a protected procedure has to be assumed to modify the protected object (neither are true in general, of course).)

> I understand that an instance of a type with default discriminants can be mutated (here I mean have its discriminant changed) via assignment. It seems like the most effective way for a compiler to support that ability is to always allocate enough memory for the object to account for all possible discriminant values. Mordechai Ben-Ari pretty much says this explicitly in his book "Ada for Software Engineers."

I personally think it was a mistake that this implementation was allowed, since it makes many useful discriminated types impossible or excessively expensive in space. Janus/Ada allocates discriminant-dependent components to size, and thus reallocates the components if a discriminant is changed. That of course has other problems (mostly in overhead if the objects are assigned often).

Most everyone disagrees with my position on this, the reasoning being that the extra memory allocation overhead isn't justified on real-time systems. Probably there is some happy medium using a combination of both implementations (clearly some real-time systems couldn't allow a reallocating implementation, but that is what we have pragma Restrictions for; for other uses, a more flexible implementation is better, IMHO).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 6 May 2010 17:15:24 +0200
Subject: Re: Question about default discriminants and mutable objects.
Newsgroups: comp.lang.ada

[…]

> I also understand that an instance of a type without default discriminants can't be mutated in this way (that is, by assignment).

You can assign it as a whole as well as its components individually. The object is mutable, but you cannot change its constraint. Compare it with similar cases::

```
S : String := "abc";
```

Here the constraint is the array bounds. They cannot be changed, yet S is mutable.

```
T : Foo'Class := ...;
```

Here the constraint is the type tag. It cannot be changed, but T is mutable.

> One could imagine some currently non-existent syntax that would allow the programmer to mark a type declaration so that the compiler allowed discriminant values to be changed via assignment without leaning on the mechanism of default discriminants.

That is not a property of the type. It is of a type constraint. Type + constraint = subtype. Ada is a bit sloppy in the object's subtype specification. When you declare S as String you write a type, but the compiler reads it as a constrained subtype of String. That is not good, but it is difficult to propose a cure. Probably:

```
S : String (<>);  -- Any subtype of String
T : Foo'Class (<>); -- Any type from
                    -- the class
```

etc.

> Furthermore one could imagine treating default discriminants as 100% syntactic sugar and not endowing them with any special semantics regarding mutability.

IMO, discriminant's defaults should imply nothing like what they do now.

From: Gene <gene.ressler@gmail.com>
Date: Wed, 5 May 2010 21:07:30 -0700 PDT
Subject: Re: Question about default discriminants and mutable objects.
Newsgroups: comp.lang.ada

[…]

I don't have any special knowledge, and I also believe there is no really good reason for the connection.

But I've always satisfied my own curiosity about this with the rationale that it's all about declarations that look like

```
x : Foo;
```

where Foo is a discriminated type.

For Foo to be well-defined, it must have a default discriminant value. Otherwise it is no type at all.

Yet the syntax here connotes that any value of type Foo ought to be assignable to x in the future. It would be an opaque gotcha if a future assignment to x with a value having other than the default discriminant caused a run-time exception.

The conclusion is that x ought to be mutable. Voila! Discriminated types with default discriminant values are mutable.

And if you're going to admit non-mutable discriminant types at all, then it's something like logical for all the rest to be these.

Tortured, I agree… One can almost hear the design-by-committee in progress.

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Wed, 5 May 2010 21:56:42 -0700 PDT
Subject: Re: Question about default discriminants and mutable objects.
Newsgroups: comp.lang.ada

Gene, you guessed quite well. See Ada 83 Rationale

http://archive.adaic.com/standards/83rat/html/ratl-04-07.html#4.7

and especially 4.7.4.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Fri, 07 May 2010 08:35:54 -0400
Subject: Re: Question about default discriminants and mutable objects.
Newsgroups: comp.lang.ada

[…]

As to default discriminants causing mutable discriminants, despite the rationale, it's a bad design. It's confusing.

And it means you can't use defaults just as defaults when you want immutable discriminants. Except you CAN do that for limited types (more confusion). And for '[in] out' parameters, you don't know at compile time whether the thing is mutable, which is just a tripping hazard.

Also, some compilers chose the deallocate/reallocate strategy, and others chose the allocate-the-max strategy. That's bad; it means you can't use the feature portably. Standards are supposed to promote uniformity.

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 6 May 2010 07:59:26 -0700 PDT
Subject: Re: Question about default discriminants and mutable objects.
Newsgroups: comp.lang.ada

[…]

> See Ada 83 Rationale

http://archive.adaic.com/standards/83rat/html/ratl-04-07.html#4.7

and especially 4.7.4.

I don't think that's a complete explanation, though. It doesn't explain why you couldn't have (1) a discriminant type without a default, in which the initial value of the discriminant must be specified for each object but the discriminant could still be changed by assigning the whole object; or (2) a discriminant type with a default discriminant but where all objects of the type are still constrained [thus making an object declaration X : T; equivalent to X : T(default);]. It's probably the inability to do one or both of those two things that causes the "annoyance" Randy mentioned.

As for why those possibilities couldn't be provided, it's probably just because they would have had to come up with some funky syntax for it. That's not a trivial consideration. Designing syntax is not always easy.

On building a GNAT cross compiler for PowerPC

From: Tom Hawkins <tomahawkins@gmail.com>
Date: Mon, 3 May 2010 08:04:28 -0700 PDT
Subject: Building GNAT
Newsgroups: comp.lang.ada

Is this an appropriate forum for discussing GNAT build problems? I am attempting to build a GNAT cross compiler targeting bare metal PowerPC platforms. After successfully getting past binutils, the first pass of GCC, and Newlib, I'm getting stuck on the final pass of GCC and GNAT. When linking gnatmake, the linker is throwing a bunch of undefined references from osint.adb (such as __gnat_is_writable_file_attr).

[…]

Many of the challenges I've had to this point have been around this use of <derint.h> and sockets, neither of which are needed on my target platform. Is there any way to configure GNAT not to build all this OS related stuff?

Recent discussions I've have with some of the issues encountered so far:

http://comments.gmane.org/gmane.comp.lib.newlib/6132

http://comments.gmane.org/gmane.comp.gcc.help/32672

[…]

From: Simon Wright <simon@pushface.org>
Date: Mon, 03 May 2010 18:03:29 +0100
Subject: Re: Building GNAT
Newsgroups: comp.lang.ada

I think your problem may be that your host compiler is an older version (4.4.3) than the source tree you're using to build the cross compiler (4.5.0). The recommendation is to build the host compiler first, then use that to build the cross.

The symbols you're missing are (in 4.5.0 but not in 4.3.4, the only handy version I have for comparison) defined in adaint.c, and it looks as though adaint.o is being picked up from the pre-built host libraries (not surprising, the cross-compiler has to run on the host so needs to use host file attributes etc).

From: Tom Hawkins <tomahawkins@gmail.com>
Date: Mon, 3 May 2010 22:03:52 -0700 PDT
Subject: Using GNAT in a C and assembly toolchain
Newsgroups: comp.lang.ada

[…]

Thanks Simon. That did the trick. I now have a functioning Ada cross compiler; I just need to learn how to use it. I could use a few pointers to get me started…

On our particular application -- an automotive ECU -- we're locked into a proprietary C-based flow. The operating system and hardware abstraction layer come precompiled and the application code must be compiled and linked with the established tool chain (GHS C, not Ada).

The application programmer defines a procedure, which is invoked by the OS at a periodic rate. This procedure references global C variables to access the hardware abstraction layer.

The question is how can I best use Ada and GNAT in such an environment? One method that comes to mind is to compile Ada to assembly code, which is then passed to GHS for assembling and linking with the closed source OS and HAL. I don't see an assembly switch, like GCC's -S. Would this mean I would need the compile Ada modules individually with GCC instead of using gnatmake? Are there any options to compile Ada down to C by chance?

And how does gnatbind fit into this picture? It appears to create initialization code. Would I just need to insert this code such that it is invoked at power up?

From: Simon Wright <simon@pushface.org>
Date: Tue, 04 May 2010 20:50:27 +0100
Subject: Re: Using GNAT in a C and assembly toolchain
Newsgroups: comp.lang.ada

[…]

> On our particular application -- an automotive ECU -- we're locked into a proprietary C-based flow. The operating system and hardware abstraction layer come precompiled and the application code must be compiled and linked with the established tool chain (GHS C, not Ada).
>
> The application programmer defines a procedure, which is invoked by the OS as a periodic rate. This procedure references global C variables to access the hardware abstraction layer.

I would have expected engine control software to require a pretty high degree of certification, if not for safety-related reasons then because fixing problems in deployed firmware is going to be expensive and possibly embarrassing, see Toyota, and part of that involves the tool chain. OK, I work on military systems, but my Software Design Authority would need A Whole Lot Of Convincing to change the tool chain in a way not supported by the tool vendor.

> The question is how can I best use Ada and GNAT in such an environment? […]

"gnatmake -c -S foo.adb" compiles the closure of foo to assembler (.s) files. Of course these are going to be in GNU assembler, which may or may not be that used by GHS.

There is a tool created by Sofcheck called AdaMagic which generates C. GCC does not.

> And how does gnatbind fit into this picture? It appears to create initialization code. Would I just need to insert this code such that it is invoked at power up?

I think you'd call gnatbind with -C (to generate C binder code rather than Ada) and -n (no Ada main program).

The generated code includes an adainit() to be called to initialize the Ada runtime and an adafinal() to finalize it (not that I've ever had to call adafinal()).

From: Tom Hawkins <tomahawkins@gmail.com>
Date: Tue, 4 May 2010 22:52:20 -0700 PDT
Subject: Re: Using GNAT in a C and assembly toolchain
Newsgroups: comp.lang.ada

[…]

> OK, I work on military systems, but my Software Design Authority would need A Whole Lot Of Convincing to change the tool chain in a way not supported by the tool vendor.

This is strictly prototyping.

[…]

> "gnatmake -c -S foo.adb" compiles the closure of foo to assembler (.s) files. Of course these are going to be in GNU assembler, which may or may not be that used by GHS.

This will work. And GHS does indeed read GNU assembly, with minor post processing.

BTW, I've moved the scripts and patch files used to build the powerpc-eabi cross compiler to github if anyone's interested:

http://github.com/tomahawkins/powerpc-eabi

On unit recompilation with GNAT

From: Lambertus Dries <ldries46@planet.nl>
Date: Fri, 9 Apr 2010 15:13:46 +020
Subject: GPS Compiler options
Newsgroups: comp.lang.ada

I'm relatively new to GPS. I'm trying to compile a program but every time I compile the program I observe that all units are compiled when I had expected that only the unit I had altered would be compiled again. What do I have to do to make this work. I have tried already all types of building of the program.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 9 Apr 2010 07:15:07 -0700 PDT
Subject: Re: GPS Compiler options
Newsgroups: comp.lang.ada

[…]

The bug may be either in gnatmake or in your expectations; this depends on which unit you change.

If you change a spec, then all specs and all bodies that depend on the changed spec need to be recompiled.

If you change a generic body, then all specs and bodies that contain an instantiation of the generic body need to be recompiled (this is specific to GNAT, which does not share the object code between instantiations of a generic).

If you change a file containing a "separate" unit, then the enclosing unit needs to be recompiled too, because GNAT emits only one object file containing the enclosing units and all "separate" bodies in it.

We could help you more if you would be more specific about the units in your program and on which units you changed.

PS. Maybe consider using the gnatmake -m switch ("minimal recompilation"); you do that by changing the project properties in GPS. With this option, gnatmake expends more effort trying to determine which units are still up to date.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 9 Apr 2010 16:26:05 +0200
Subject: Re: GPS Compiler options
Newsgroups: comp.lang.ada

[…]

Also, when the object and ali-files are on a remote server, usually together with the sources (this is the case at my work), accessed over a network file system (samba, NFS etc), then if the compiling computer has the clock unsynchronized with the server, you may get the described effect (or worse, non-compiled files).

From: Simon J. Wright <simon.j.wright@mac.com>
Date: Fri, 9 Apr 2010 08:10:34 -0700 PDT
Subject: Re: GPS Compiler options
Newsgroups: comp.lang.ada

And (experience on a previous project with Solaris NFS) much longer compilation times.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 9 Apr 2010 09:40:48 -0700 PDT
Subject: Re: GPS Compiler options
Newsgroups: comp.lang.ada

> Also, when the object and ali-files are on a remote server, […] you may get the described effect (or worse, non-compiled files).

I think gnatmake -m corrects this problem. With this option, gnatmake no longer relies on the timestamps but only on the CRC32 values in the .ali files.

From: Simon J. Wright <simon.j.wright@mac.com>
Date: Fri, 09 Apr 2010 18:20:57 +0100
Subject: Re: GPS Compiler options
Newsgroups: comp.lang.ada

[…]

I believe that it first compares the timestamps: if the same, no change, but if different, does the ali check.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 9 Apr 2010 10:46:11 -0700 PDT
Subject: Re: GPS Compiler options
Newsgroups: comp.lang.ada

[…]

Exactly. And gnatmake will then do the right thing (i.e. not recompile) if the timestamps are different only because the clocks of various machines are out of sync.

From: Simon J. Wright <simon.j.wright@mac.com>
Date: Fri, 09 Apr 2010 22:00:58 +0100
Subject: Re: GPS Compiler options
Newsgroups: comp.lang.ada

Indeed. But it takes much longer, because it has to open each file; and this is even worse over a network.

GNAT 3.16a1 on Windows had a wonderful feature where the RTS timestamps were out by an hour unless you installed in daylight savings time! This led to opening all the relevant RTS files, every time, every compilation -- and to an interesting work instruction.

SPARK: what does it prove?

From: Peter C. Chapin
<pcc482719@gmail.com>
Date: Fri, 28 May 2010 09:25:50 -0400
Subject: SPARK: What does it prove?
Newsgroups: comp.lang.ada

There has been a lot of discussion about SPARK on this group recently. That's great, but I hope those who are more interested in full Ada aren't getting annoyed! :)

It is common to talk about SPARK proofs but of course what the Simplifier is actually proving are the verification conditions generated by the Examiner.

Formally this leaves open the question of if those verification conditions have anything to do with reality or not. Ultimately, it seems to me, before one can formally prove anything about the behavior of a program one needs a formal semantics for the programming language in question. It is my understanding that SPARK95 does not have a formal semantics. Thus the Examiner is producing VCs based on the informal description of Ada in the reference manual. What if that information description is, as many such descriptions are, logically inconsistent or ambiguous? I realize that SPARK is intended to restrict the Ada language to remove ambiguity and implementation specific behavior, but is there a proof that it actually does?

Without a formal semantics of SPARK, then it seems like the "proofs" produced by the tools are not really proving anything… in a mathematically rigorous sense at least. I guess this is why Praxis calls SPARK a semi-formal method.

I understand that the real goals of SPARK are to help practitioners produce reliable software… not generate rigorous proofs just for the sake of doing so. To that end, following the informal specification of Ada in the reference manual seems perfectly reasonable. The features of Ada that SPARK retains are simple with (mostly) "obvious" semantics, so why quibble over every mathematical detail? I'm fine with that. The tools *do* help me write more reliable programs and that's great!

Still it would be more satisfying if there was a formal semantics for SPARK to "back up" what the tools are doing. I actually read an article recently about programming language semantics that mentioned (is this true?) that one of the original requirements in the development of Ada was the production of a formal semantics for Ada. I even understand that there were two attempts to produce such a semantics. Here are those references:

1. V. Donezeau-Gouge, G. Kahn, and B. Lang. On the formal definition of Ada. In Semantics-Directed Compiler Generation, Lecture Notes in Computer Science, vol 94, pp 475-489, Springer, Berlin, 1980

2. D. Bjorner and O.N. Oest. Towards a Formal Description of Ada, Lecture Notes in Computer Science, vol 98, Springer, Berlin 1980.

The article I'm reading is "Programming Language Description Languages" by Peter D. Moses in the book "Formal Methods: State of the Art and New Directions" edited by Paul P. Boca, Janathan P. Bowen, and Jawed I. Siddiqi, published by Springer, (C) 2010.

I understand that the efforts above were incomplete and even then only apply to Ada 83. I also understand that few full scale languages have a formal semantics (do any?). It seems a shame, though, that Ada does not have one considering especially the way Ada is used.

[…]

From: Rod Chapman
<roderick.chapman@googlemail.com>
Date: Fri, 28 May 2010 06:55:18 -0700 PDT
Subject: Re: SPARK: What does it prove?
Newsgroups: comp.lang.ada

[…]

> It is common to talk about SPARK proofs but of course what the Simplifier is actually proving are the verification conditions generated by the Examiner.
>
> Formally this leaves open the question of if those verification conditions have anything to do with reality or not. Ultimately, it seems to me, before one can formally prove anything about the behavior of a program one needs a formal semantics for the programming language in question. It is my understanding that SPARK95 does not have a formal semantics.

Well..not quite. The VC generator was constructed and very much based on the formal semantics for SPARK83 that was constructed in the mid-1990s. We have _lots_ of confidence that this semantics is completely upwards-compatible and consistent with the canonical semantics of Ada95 and thus SPARK95 - SPARK2005.

Unfortunately, we did not have the funding to keep that SPARK83 semantics up to date with new features that were added later like modular types from Ada95, but these are a fairly modest extension to the language.

There are also lots of assumptions that underlie any "proof" of anything - in our case the integrity of the compiler, linker and the rest of the build environment, the implementation of the target processor itself and many other things. While these are valid concerns, these assumptions really do seem to hold up in the "real world" - i.e. with our customers using real commercial compilers, microprocessors and so on.

In short: it seems to work.

[…]

From: Peter C. Chapin
<pcc482719@gmail.com>
Date: Fri, 28 May 2010 11:58:18 -0400
Subject: Re: SPARK: What does it prove?
Newsgroups: comp.lang.ada

> Well..not quite. The VC generator was constructed and very much based on the formal semantics for SPARK83 that was constructed in the mid-1990s. We have _lots_ of confidence that this semantics is completely upwards-compatible and consistent with the canonical semantics of Ada95 and thus SPARK95 - SPARK2005.

That's interesting to know. Thanks.

> There are also lots of assumptions that underlie any "proof" of anything - in our case the integrity of the compiler, linker and the rest of the build environment...

Yes, definitely. This was a point I tried to make in a different thread related to testing SPARK programs (and the value of doing so). SPARK helps show that the source code is in some sense correct, which is great, but that's not the whole story.

> In short: it seems to work.

Absolutely. I hope you didn't take my post as a criticism of SPARK. If the goal is to reduce errors in actual deployed programs, which at the end of the day is the important thing it seems, then I agree that SPARK works!

From: Jeffrey R. Carter
<jrcarter@acm.org>
Date: Mon, 31 May 2010 16:36:53 -0700 PDT
Subject: Re: SPARK: What does it prove?
Newsgroups: comp.lang.ada

[…]

> 1. V. Donezeau-Gouge, G. Kahn, and B. Lang. On the formal definition of Ada.
>
> In Semantics-Directed Compiler Generation, Lecture Notes in Computer Science, vol 94, pp 475-489, Springer, Berlin, 1980

> 2. D. Bjorner and O.N. Oest. Towards a Formal Description of Ada, Lecture Notes in Computer Science, vol 98, Springer, Berlin 1980.
>
> I understand that the efforts above were incomplete and even then only apply to Ada 83. […]

I think you're mistaken: these applied only to Ada 80 (MIL-STD-1815).

From: Yannick Duchêne
<yannick_duchene@yahoo.fr>
Date: Mon, 31 May 2010 03:17:32 +0200

Subject: Re: SPARK: What does it prove?
Newsgroups: comp.lang.ada

The question, “what does it prove ?”, raise another corollary question, which is “what can it say ?” or “what can it talk about ?”.

I'm currently trying to make proofs on some binary stuffs, which has always seems obvious to me, and at the time of trying to prove it, I see I can't even prove the third of the initial post-condition I wanted my functions to have, because there are some I can't prove at all (I'm not talking about RTC, rather about post-condition expressing properties, and it is far less easy than proving RTC conditions).

If is funny to note that these difficulty are a consequence of SPARK tied to Ada.

An example : Ada has modular type, but can't see modular types has polynomials, and the relevant modal, which could help, would be this one : polynomial. Ada does not have this, SPARK too doesn't.

Another things also : sometime, it is better to make a proof on an abstract algorithm, which is not efficient, as it is too much difficult to apply the same proof (prove post-conditions from preconditions and the algorithm) with the efficient version. However, it would be more easy to demonstrate that the efficient algorithm is an equivalent transformation of the more abstract non-efficient one.

I mean, prove something on function F, demonstrate function G is equivalent to function F, so as legally assert the post-conditions of F are also prove on G, because there was on F and G is equivalent to F.

This is another kind of thing SPARK cannot express or talk/say about.

This may be the start of some answers to the question “what can it prove ?” or “what can't it prove ?”, which are similar questions.

From: Phil Thornley <phil.jpthornley@googlemail.com>
Date: Mon, 31 May 2010 06:05:04 -0700 PDT
Subject: Re: SPARK: What does it prove?
Newsgroups: comp.lang.ada

[…]

> I mean, prove something on function F, demonstrate function G is equivalent to function F, so as legally assert the postconditions of F are also prove on G, because there was on F and G is equivalent to F.
>
> This is another kind of thing SPARK cannot express or talk/say about.

How about using proof abstraction? Put one set of post-conditions (for the inefficient version) on the spec and the other set (for the efficient version) on the body.

Then the post-conditions on the body are proved from the code and the post-conditions on the spec are proved by a user rule that is justified by the 'offline' proof of equivalence of the two algorithms.

*** BUT *** the current GPL version (8.1.1) sometimes gets that post-condition refinement VC wrong. This only seems to happen when there is a refined pre and post-condition on the body but no refined state data, eg for private types, which is where I came across the problem.

(Notified to report@gnat.com on 9th February).

On loop assertions in SPARK

From: Peter C. Chapin <pcc482719@gmail.com>
Date: Sat, 06 Mar 2010 15:33:12 -0500
Subject: Trouble cutting loops with SPARK. Advice?
Newsgroups: comp.lang.ada

I'm attempting to use SPARK to prove a simple function is free of run time errors. Here is the (simplified) package specification:

```
package Strings is
  Max_Line_Length: constant := 225;
  subtype Line_Index is Positive
    range 1 .. Max_Line_Length;
  subtype Line_Size_Type is Natural
    range 0 .. Max_Line_Length;
  subtype Line_Type is
    String(Line_Index);
  type Line_Record is
    record
      Text : Line_Type;
      Size : Line_Size_Type;
    end record;
  function Prepare_Line(Line : String)
    return Line_Record;
end Strings;
```

This package provides a type Line_Record that holds a variable length string in a fixed size buffer. As you can see the lines are limited to 225 characters. The function Prepare_Line copies the given string into a Line_Record and returns it. If the string is too long it is truncated.

Okay… here is the implementation:

```
package body Strings is
  function Prepare_Line(Line : String)
      return Line_Record is
    Result : Line_Record;
    Characters_To_Copy:
                    Line_Size_Type;
  begin
    Result.Text :=
              Line_Type'(others => ' ');
    if Line'Length > Max_Line_Length
        then
      Characters_To_Copy :=
              Max_Line_Length;
    else
      Characters_To_Copy :=
              Line'Length;
    end if;
    for I in Line_Size_Type range
        1 .. Characters_To_Copy loop
      Result.Text(I) :=
              Line(Line'First + (I - 1));
    end loop;
    Result.Size := Characters_To_Copy;
    return Result;
  end Prepare_Line;
end Strings;
```

The problem is with the loop that actually copies the characters from the provided String into the Line_Record. SPARK generates a verification condition that it can't prove related to the "run time check" on the line in the body of the loop. The SPARK Examiner warns about using a default assertion to cut the loop. I understand that I should probably provide my own assertion. My problem is that I can't quite figure out what I need to say.

If I write an assertion in the loop that involves the loop index variable, I, the generated verification conditions appear to try to prove that the assertion holds even for one extra iteration. That is, there is a VC that tries to prove that if the assertion is true, then it must be true for the next iteration. However that does not need to be so on the last iteration (since there is no next iteration in that case) so the condition can't be proved.

Specifically I get conclusions like "I <= 224." That can't be proved because, in fact, sometimes I = 225 (the last time through the loop in the case where the source String is being truncated).

Okay… but if I stay away from talking about I in the assertion, what can I assert that will help convince SPARK that the array access operations are fine? I've tried a few things… I can be more specific if necessary. So far, no joy.

I get the feeling there is a trick here that I'm overlooking. Is my code wrong? It looks okay to me. :)

From: Phil Thornley <phil.jpthornley@googlemail.com>
Date: Sun, 7 Mar 2010 02:22:43 -0800 PST
Subject: Re: Trouble cutting loops with SPARK. Advice?
Newsgroups: comp.lang.ada

[…]

Your code is OK and yes there is something that you are probably overlooking.

It's not really a 'trick' - more an oddity of the language for proving code in 'for' loops that have range constraints. (Loops like that are the one exception to the SPARK rule that there are no anonymous subtypes.)

The problem is that the value of the variables that provide the loop bounds (in your code the variable Characters_To_Copy) can be changed within the loop, so any reference to Characters_To_Copy within an assertion in the loop means the *current* value, not the value that defined the loop bound. The Examiner doesn't check to see whether Characters_To_Copy is changed within the loop (it just assumes that it might be changed), so in this case you have to state that it is unchanged in the loop assertion.

SPARK therefore supplies a notation for the value of any variable on entry to a loop - append % to the name - so Characters_To_Copy% within a proof context in the loop means the value of Characters_To_Copy on entry to the loop.

The loop assertion that works for the code in your message is:

```
--# assert I >= 1
--# and  Characters_To_Copy
     <= Line'Length
--# and  Characters_To_Copy
     = Characters_To_Copy%;
```

placed at the beginning of the loop.

[…]

SPARK_IO in SPARK 9 & Ada 2005

From: Alexandre Konieczny <alexandre.konieczny@gmail.com>
Date: Thu, 8 Apr 2010 05:46:48 -0700 PDT
Subject: Spark 9 & Ada 2005
Newsgroups: comp.lang.ada

[…]

This problem is intended to people who know SPARK.

I'm trying to compile the Spark_IO.adb file using -gnat05 option, but there is an error.

The error is:

```
spark_io.adb:196:42: (Ada 2005) cannot copy object of a limited type (RM-2005 6.5(5.5/2))
```

This is a restriction of Ada 2005, and I saw in a message of last year saying that Spark_IO is not compatible with Ada 2005.

Is it still the case? However, the Spark Pro version 9 considers Ada 2005 features.

Does anyone know about this problem ?

From: Rod Chapman <roderick.chapman@googlemail.com>
Date: Thu, 8 Apr 2010 08:41:53 -0700 PDT
Subject: Re: Spark 9 & Ada 2005
Newsgroups: comp.lang.ada

[…]

Good question. The specification of SPARK_IO that ships with SPARK Pro 9 is compatible with both SPARK95 and SPARK2005.

The body is Ada95 (not SPARK at all), but - as you point out - is not compatible with GNAT in Ada2005 mode. We will raise an internal ticket here to produce an Ada2005-friendly body for SPARK_IO for a future release.

Thanks for bringing this to our attention.

On the accessibility level of access discriminants

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 6 Apr 2010 14:17:04 -0700 PDT
Subject: Language lawyer question: access discriminants
Newsgroups: comp.lang.ada

I'm hoping someone who understands the rules about access discriminant accessibility level can answer this definitively. This is a reduced example of something I found in someone else's code:

```
package Pack1 is
   type Rec is record
      F1 : Integer;
   end record;
   type Rec2 (D : access Rec)
         is limited record
      F2 : Integer;
   end record;
   function Func (Param : Integer)
         return Rec2;
end Pack1;

with Pack1;  use Pack1;
procedure Proc2 is
   A : access Rec;
begin
   A := Func(1).D;   -- LEGAL?
end Proc2;
```

I think the statement marked LEGAL? is illegal, because of rules saying that the result of Func is an object inside a nested master that consists of just the one assignment statement, and the accessibility level of the access discriminant (Func(1).D) is the accessibility level of the enclosing object (the temporary object containing the result of Func), and therefore the accessibility level of Func(1).D is deeper than that of A. But the rules are pretty complex and I'm hoping someone in the know can straighten me out if I'm wrong. (And I'm not interested in any replies that say "XYZ compiler says it's legal". I need to know what the standard says.)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 7 Apr 2010 15:04:31 -0500
Subject: Re: Language lawyer question: access discriminants
Newsgroups: comp.lang.ada

[…]

This program is depending on the accessibility of access discriminants of an object returned from a function. Those are not currently defined. We know that the rules as defined in Ada 2005 don't work (have various nasty holes). The rules corrections proposed by AI05-0051-1 have never been approved, and so far as I know, are only understood by the author (Tucker Taft).

My (lousy) understanding of the rules proposed in AI05-0051-1 is that the accessibility of the access discriminants of the returned object are that of the call site. The rules Tucker proposed would generally cause the accessibility of the LHS of the assignment to propagate into the function, and a (dynamic) check would be made at the return statement that the accessibility is sufficient.

The net effect is that this call is always legal, but it is very likely that it would fail an accessibility check at the return statement -- thus it would usually raise Program_Error.

> I think the statement marked LEGAL? is illegal, because of rules saying that the result of Func is an object inside a nested master that consists of just the one assignment statement, and the accessibility level of the access discriminant (Func(1).D) is the accessibility level of the enclosing object (the temporary object containing the result of Func), and therefore the accessibility level of Func(1).D is deeper than that of A. […]

What I can't say for sure is whether the accessibility of the LHS would be used or whether that of a assignment statement would be used. I'd have to go and read the AI again several times and even then I wouldn't really know for sure. I do know that type conversions get the right accessibility (so using a named type here and a type conversion would surely be legal).

But remember this is all based on the unapproved binding interpretation AI05-0051-1. What the standard says currently is garbage, and it isn't worth figuring out what those rules are (no one will ever try to enforce those - that is, no ACATS test will ever exist for the rules as currently written).

So my best advice is do whatever you want now, and plan to adjust to AI05-0051-1 rules when (if?) that gets finished.

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

July 01-02 — 4th **International Workshop on Verification and Evaluation of Computer and Communication Systems** (VECoS'2010), Paris, France. Topics include: Model-checking; Abstraction techniques; Performance and robustness evaluation; Dependability assessment techniques; QoS evaluation, planning and deployment; etc.

July 01-04 — 20th **International Workshop on Algebraic Development Techniques** (WADT'2010), Etelsen, Germany. Topics include: Other approaches to formal specification; Specification languages, methods, and environments; Model-driven development; Integration of formal specification techniques; Formal testing and quality assurance, validation, and verification; etc.

July 05-09 — 7th **International Summer School on Software Engineering** (ISSSE'2010), Salerno, Italy. Topics include: Model Driven Software Engineering, Empirical Software Engineeering, Software Evolution, Program Comprehension, etc.

July 05-12 — 37th **International Colloquium on Automata, Languages and Programming** (ICALP'2010), Bordeaux, France. Topics include: Parallel and Distributed Computing; Principles of Programming Languages; Formal Methods and Model Checking; Models of Concurrent and Distributed Systems; Models of Reactive Systems; Program Analysis and Transformation; Specification, Refinement and Verification; Type Systems and Theory; etc.

☺ July 07-09 — 9th **International Symposium on Parallel and Distributed Computing** (ISPDC'2010), Istanbul, Turkey. Topics include: Parallel Computing; Distributed Systems Methodology and Networking; Parallel Programming Paradigms and APIs; Tools and Environments for Parallel Program Analysis; Task Scheduling and Load Balancing; Performance Management in Parallel and Distributed Systems; Distributed Software Components; Real-time Distributed and Parallel Systems; Security in Parallel and Distributed Systems; Fault Tolerance in Parallel and Distributed Systems; Parallel Scientific Computing and Large Scale Simulations; Parallel and Distributed Applications; etc.

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, such as Component-based software engineering, Critical software engineering, Distributed and parallel software architectures, Education aspects of software engineering, Embedded software engineering, Empirical software engineering, Model Driven Architecture (MDA), Model-oriented software engineering, Object-oriented methodologies, Performance critical systems, Program understanding, Programming languages, Quality issues, Real-time software engineering, Real-time software systems, Reliability, Reverse engineering, Software architectures and design, Software design patterns, Software maintenance, Software reuse, Software safety and reliability, Software security, Software specification, Software tools, Verification and validation of software, etc.

July 13-14 — 2nd **Software Engineering Method and Theory Workshop** (SEMAT'2010), Washington DC, USA. Topics include: the prevalence of fads more typical of fashion industry than of an engineering discipline; the lack of a sound, widely accepted theoretical basis; the huge number of methods and method variants, with differences little understood and artificially magnified; the lack of credible experimental evaluation and validation; the split between industry practice and academic research; etc.

July 14-15 — 10th **International Conference on Quality Software** (QSIC'2010), Zhangjiajie, China. Topics include: Software quality (review, inspection and walkthrough, reliability, safety and security, ...); Evaluation of software products and components (static and dynamic analysis, validation and verification); Information and knowledge management (economics of software quality, ...); Formal methods (program analysis, ...); Applications (component-based systems, distributed systems, embedded systems, enterprise applications, information systems, safety critical systems, ...); etc.

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.

> ☺ July 20-21 — **Workshop on Exploiting Concurrency Efficiently and Correctly** ((EC)^2). Topics include: deficiencies in current languages and tools; multi-core software design, correctness issues, and correctness approaches; programming languages and paradigms that facilitate concurrency exploitation; novel approaches for teaching concurrency; significant case studies; etc.

July 19-23 — COMPSAC2010 - 4th IEEE **International Workshop on Quality Oriented Reuse of Software** (QUORS'2010), Seoul, South Korea. Topics include: High quality software reuse methods; Dependable Component-Based Systems; Software product lines; Quality aspects of design patterns; Software evolution; Model-driven software engineering; Component and service repository; Reuse in Embedded Software Systems; Case studies and experience reports.

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.

July 25-28 — 29th Annual ACM SIGACT-SIGOPS **Symposium on Principles of Distibuted 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.

☺ August 23-25 — 16th IEEE **International Conference on Embedded and Real-Time Computing Systems and Applications** (RTCSA'2010), Macau SAR, P.R.China. Topics include: Software design for heterogeneous multi-core embedded platform, Multi-thread programming for multi-core embedded platform, Embedded system design practices, Real-time scheduling, Timing analysis, Programming languages and run-time systems, Middleware systems, Design and analysis tools, Case studies and applications, etc.

August 23-26 — 5th IEEE **International Conference on Global Software Engineering** (ICGSE'2010), Princeton, NJ, USA. Topics include: Strategic issues in distributed development: cost-benefit-risk analysis, ...; Methods and tools for distributed software development: requirements engineering, design, coding, verification, testing and maintenance, development governance; Empirical studies and lessons learnt on distributed development; etc.

☺ Aug 31 – Sep 09 — 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.

September 01-03 — 7th **International Colloquium on Theoretical Aspects of Computing** (ICTAC'2010), Natal, Rio Grande do Norte, Brazil. Topics include: principles and semantics of programming languages; software specification, refinement, verification; integration of theories, formal methods and tools for engineering computing systems; models of concurrency and security; theory of parallel, distributed, and grid computing; real-time and embedded systems; case studies, theories, tools and experiments of verified systems; domain-specific modeling and technology; etc.

☺ Sep 11-15 — 19th **International Conference on Parallel Architectures and Compilation Techniques** (PACT'2010), Vienna, Austria. Topics include: ground-breaking research related to parallel systems ranging across instruction-level parallelism, thread-level parallelism, multiprocessor parallelism and large scale systems, such as Parallel computational models; Compilers and tools for parallel computer systems; Support for concurrency correctness in hardware and software; Parallel programming languages, algorithms and applications; Middleware and run-time system support for parallel computing; Reliability and fault tolerance for parallel systems; Modeling and simulation of parallel systems and applications; Parallel applications and experimental systems studies; Case studies of parallel systems and applications; etc.

September 12-18 26th IEEE **International Conference on Software Maintenance** (ICSM'2010), Timisoara, Romania.

Sep 12-13 6th **International Conference on Predictive Models in Software Engineering** (PROMISE'2010). Topics include: Industrial experience reports detailing the application of software technologies - processes, methods, or tools - and their effectiveness in industrial settings; Tools for software researchers that effectively gather and analyze data to support reproducible and verifiable research; etc.

☺ Sep 13-16 39th **International Conference on Parallel Processing** (ICPP'2010), San Diego, California, USA. Topics include: compilers and languages, etc.

September 17 SEFM2010 - 2nd **Workshop on Formal Methods and Agile Methods** (FM+AM'2010), Pisa, Italy. Topics include: novel contributions that can be used for making rapid development techniques more formally sound, as well as for accelerating the speed of formally sound development techniques. Deadline for early registration: July 30, 2010.

September 20-24 25th IEEE/ACM **International Conference on Automated Software Engineering** (ASE'2010), Antwerp, Belgium. Topics include: Component-based systems; Maintenance and evolution; Model-based software development; Model-driven engineering and model transformation; Modeling language semantics; Open systems development; Product line architectures; Program understanding; Program transformation; Re-engineering; Specification languages; Software architecture and design; Testing, verification, and validation; etc.

☺ Sep 20 3rd **International Workshop on Academic Software Development Tools** (WASDeTT'2010). Topics include: How to integrate and combine independently developed tools? What are the positive lessons learned and pitfalls in building tools? What are effective techniques to improve the quality of academic tools? What particular languages and paradigms are suited to build tools?

Sep 20 7th **International Workshop on Model-based Methodologies for Pervasive and Embedded Software** (MOMPES'2010).

Sep 20-21 Joint ERCIM **Workshop on Software Evolution and International Workshop on Principles of Software Evolution** (IWPSE-EVOL'2010). Topics include: Application areas: distributed, embedded, real-time, ultra large scale, information systems, ...; Technical aspects: co-evolution and inconsistency management, impact analysis and change propagation, languages and notations for supporting evolution, ...; Managerial aspects: risk analysis, software quality, productivity, training, ...; Empirical studies related to software evolution; Industrial experience on successes and failures related to software evolution;

☺ Sep 20-21 15th **International Workshop on Formal Methods for Industrial Critical Systems** (FMICS'2010), Antwerp, Belgium. Topics include: Design, specification, code generation and testing based on formal methods; Verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability; Tools for the development of formal design descriptions; Case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; Impact of the adoption of formal methods on the development process and associated costs; Application of formal methods in standardization and industrial forums; etc.

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.

☺ Sep 27-29 CBSoft'2010 - 14th **Brazilian Symposium on Programming Languages** (SBLP'2010), Salvador, Bahia, Brazil. Topics include: Programming language design and implementation, Design and implementation of programming language environments, Object-oriented programming languages, Program transformations, Program analysis and verification, Compilation techniques, etc.

Sep 29 – Oct 01 5th **International Conference on Graph Transformation** (ICGT'2010), Enschede, The Netherlands. Topics include: languages, tool support and applications in: Software architecture; Software quality, testing and evolution; Model-driven development, especially model transformations; Implementation of programming languages; Massively parallel computing; etc.

Sep 30–Oct 1 9th **International Workshop on Parallel and Distributed Methods in verifiCation** (PDMC'2010). Topics include: all aspects related to the verification and analysis of very large and complex systems using methods and techniques that exploit state-of-the-art hardware architectures, such as multi-core model checking, distributed model checking, parallel/distributed theorem proving, tools and case studies, industrial applications, etc.

Sep 29 – Oct 01 9th **International Conference on Software Methodologies, Tools and Techniques** (SoMeT'2010), Yokohama, Japan. Topics include: Software methodologies, and tools for robust, reliable, non fragile software design; Software developments techniques and legacy systems; Automatic software generation versus reuse, and legacy systems; Intelligent software systems design, and software evolution techniques; Agile Software and Lean Methods; Software optimization and formal methods for software design; Software maintenance; Software security tools and techniques, and related Software Engineering models; Formal techniques for software representation, software testing and validation; Software reliability, and software diagnosis systems; Model Driven Development (DVD), code centric to model centric software engineering; etc.

☺ October 03-05 MoDELS2010 - 3rd **International Workshop on Model Based Architecting and Construction of Embedded Systems** (ACES-MB'2010), Oslo, Norway. Topics include: model-oriented counterparts, together with the related analysis and development methods, of languages with particularly well-behaved semantics, such as synchronous languages and models (Lustre/SCADE, Signal/Polychrony, Esterel), super-synchronous models (TTA, Giotto), scheduling-friendly models (HRT-UML, Ada Ravenscar), etc.

October 06-07 5th **International Workshop on Systems Software Verification** (SSV'2010), Vancouver, Canada. Theme: "Real Software, Real Problems, Real Solutions". Topics include: static analysis, model-driven development, embedded systems development, programming languages, verifying compilers, software certification, software tools, experience reports, etc.

October 10-13 9th **International Conference on Generative Programming and Component Engineering** (GPCE'2010), Eindhoven, The Netherlands. Topics include: Generative techniques for Product-line architectures, Distributed, real-time and embedded systems, Model-driven development and architecture, Safety critical systems; Component-based software engineering (Reuse, distributed platforms and middleware, distributed systems, evolution, patterns, development methods, formal methods, etc.); Integration of generative and component-based approaches; Industrial applications; etc.

October 12-13 3rd **International Conference on Software Language Engineering** (SLE'2010). Topics include: Formalisms used in designing and specifying languages and tools that analyze such language descriptions; Language implementation techniques; Program and model transformation tools; Language evolution; Approaches to elicitation, specification, or verification of requirements for software languages; Design challenges in SLE; Applications of languages including innovative domain-specific languages or "little" languages; etc.

☺ October 17-20 25th **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2010), Reno/Tahoe, Nevada, USA. Topics include: all aspects of programming languages and software engineering, broadly construed; any aspect of software development, including requirements, modeling, prototyping, design, implementation, generation, analysis, verification, testing, evaluation, project cancellation, maintenance, reuse, regeneration, replacement, and retirement of software systems; tools (such as new programming languages, dynamic or static program analyses, compilers, and garbage collectors) or techniques (such as new programming methodologies, type systems, design processes, code organization approaches, and management techniques) designed to reduce the time, effort, and/or cost of software systems.

♦ Oct 24-28 ACM SIGAda **Annual International Conference on Ada and Related Technologies** (SIGAda'2010), Fairfax, Virginia, USA (a suburb of Washington, DC). Sponsored by ACM SIGAda, in cooperation with SIGBED, SIGCAS, SIGCSE, SIGPLAN, Ada-Europe, and the Ada Resource Association.

October 25-29	Grid2010 - **Workshop on Component-Based High Performance Computing** (CBHPC'2010), Brussels, Belgium. Topics include: Programming environments and paradigms, Analysis and comparison of existing programming approaches, Tools and Environments for Coupling of Parallel Application codes, etc.
October 25-29	14th IEEE **International Enterprise Computing Conference** (EDOC'2010), Vitória, ES, Brazil. Topics include: Organization and principles of software factories; State of the art in distributed enterprise applications; Industry specific solutions, e.g. for aerospace, automotive, finance, etc.
☺ November 01-03	29th IEEE **International Symposium on Reliable Distributed Systems** (SRDS'2010), Delhi, India. Topics include: security, safety-critical systems and critical infrastructures, fault-tolerance in embedded systems, analytical or experimental evaluations of dependable distributed systems, formal methods and foundations for dependable distributed computing, etc.
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.
☺ December 07-11	16th IEEE **International Conference on Parallel and Distributed Systems** (ICPADS'2010), Shanghai, China. Topics include: Parallel and Distributed Algorithms and Applications, Multi-core and Multithreaded Architectures, Resource Management and Scheduling, Security, Dependable and Trustworthy Systems, Real-Time Systems, Embedded systems, etc.
☺ December 08-11	11th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2010), Wuhan, China. Topics include: all areas of parallel and distributed computing.
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

2011

☺ February 09-10	3rd **International Symposium on Engineering Secure Software and Systems** (ESSoS'2011), Madrid, Spain. Topics include: security architecture and design for software and systems; verification techniques for security properties; systematic support for security best practices; programming paradigms for security; processes for the development of secure software and systems; etc. Deadline for submissions: September 13, 2010 (abstracts), September 20, 2010 (papers, tutorials).
☺ March 21-25	SAC2011 - **Track on Object-Oriented Programming Languages and Systems** (OOPS'2011), TaiChung, Taiwan. Topics include: Language design and implementation; Type systems, static analysis, formal methods; Integration with other paradigms; Aspects, components, and modularity; Distributed, concurrent or parallel systems; Interoperability, versioning and software adaptation; etc. Deadline for submissions: August 24, 2010 (full papers).
☺ April 10-13	6th **European Conference on Computer Systems** (EuroSys'2011), Salzburg, Austria. Topics include: all areas of operating systems and distributed systems, including systems aspects of Dependable computing and storage, Distributed computing, Parallel and concurrent computing, Programming-language support, Real-time and embedded computing, Security, etc. Deadline for submissions: October 3, 2010 (abstracts), October 10, 2010 (full papers).
April 12-15	2nd **International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering** (PARENG'2011), Ajaccio, Corsica, France.
April 20-24	17th **International Symposium on Formal Methods** (FM'2011), Limerick, Ireland.
☺ May 16-20	25th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2011), Anchorage, Alaska, USA. Topics include: all areas of parallel and distributed processing, such as: Parallel and distributed algorithms; Applications of parallel and distributed computing; Parallel and distributed

	software, including parallel and multicore programming languages and compilers, runtime systems, middleware, libraries, parallel programming paradigms, programming environments and tools, etc. Deadline for submissions: September 24, 2010 (abstracts), October 1, 2010 (papers).
☺ May 21-28	33rd **International Conference on Software Engineering** (ICSE'2011), Waikiki, Honolulu, Hawaii, USA. Theme: "Software by Design".
June 20-23	2011 **International Conference for Computational Science and its Applications** (ICCSA'2011), Santander, Spain. Deadline for submissions: July 31, 2010 (workshops, technical sessions), December 31, 2010 (abstracts, full papers).
♦ June 20-24	16th **International Conference on Reliable Software Technologies - Ada-Europe'2011**, Edinburgh, UK. Co-located with the **Ada Conference UK 2011**, organized under the common name of "**The Ada Connection**". Sponsored by Ada-Europe, in cooperation with ACM SIGAda (approval pending). Deadline for submissions: November 21, 2010 (papers, tutorials, workshops), January 8, 2011 (industrial presentations).
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

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

Udvar-Hazy Space Museum
(close to Dulles Airport and the conference hotel)

Conference Hotel, Hyatt Fair Lakes, Fairfax, VA 22033 (USA)

Advance Program coming in July 2010. For details, visit

http://www.sigada.org/conf/sigada2010/

Conference Highlights

Keynote Address: Systems Software Integrity Assurance to FAA's Next Generation (NextGen) Constituents

Chris Lane, Lockheed Martin Corporation

Data Comm is a program that will enhance existing communications between the air traffic controller and the pilot by essentially sending digital messages to supplement the existing voice communications. With more reliance on Data Comm as the FAA's Next Generation systems become fielded, ensuring the communications is reliable, accurate, and most importantly safe becomes increasingly critical. RTCA DO-278 provides the guidelines for communications, navigation, surveillance, and air traffic management systems software integrity assurance. It doesn't guarantee that the software developed in accordance with these guidelines is safe but if followed it ensures that the processes are in place to properly plan, develop and verify the software. Lockheed Martin is in the process of integrating Data Comm with the En Route Automation and Modernization (ERAM) program and is developing the program in compliance with DO-278. This brings challenges as well as opportunities with the increasing reliance on commercial off the shelf (COTS) software. These challenges and some insight into developing systems to the standards of DO-278 will be discussed.

Tutorial: Developing Unmanned Systems in Ada over RTEMS

Cindy Cicalese et al, The MITRE Corporation

This hands-on tutorial provides an introduction to the development of software in Ada for unmanned systems. The authors will demonstrate how they are using Ada over RTEMS in developing the real time control software for a large unmanned ground vehicle. RTEMS is an open source, real-time operating system that provides a high performance environment for embedded applications on a range of processors and embedded hardware.

Workshop on Software Security

Stephen Michell, Maurya Software, Inc., Ottawa, Ontario, Canada

This hands-on workshop will involve a "find the vulnerability" exercise in sample code, using examples from C, C++, scripting languages and Ada.

Update on the Forthcoming Ada 2012 Standard

Ed Schonberg, AdaCore, Inc.

The Ada Rapporteur Group (ARG) is nearing completion of its work on the forthcoming standard for Ada 2012. This update will present the state of the ARG work and an informal discussion of the language enhancements in the new standard.

Advance Program coming in July 2010. For details, visit

http://www.sigada.org/conf/sigada2010/

Preliminary Call for Papers

The Ada Connection

16th International Conference on Reliable Software Technologies - Ada-Europe 2011

Ada Conference UK 2011

20 - 24 June 2011, Edinburgh, UK

http://www.ada-europe.org/conference2011

General Information

The Ada Connection combines the 16th International Conference on Reliable Software Technologies – *Ada-Europe 2011* – with *Ada Conference UK 2011*. It will take place in Edinburgh, Scotland's capital city and the UK's most popular conference destination.

In traditional Ada-Europe style, the conference will span a full week, including a three-day technical program and vendor exhibition from Tuesday to Thursday, along with parallel tutorials and workshops on Monday and Friday. The Ada Connection will also encompass technical and vendor tracks under the banner of Ada Conference UK, which exists to promote awareness of Ada and to highlight the increased relevance of Ada in safety- and security-critical programming.

The Ada Connection will thus provide a unique opportunity for interaction and collaboration between academics and industrial practitioners.

Schedule

21 November 2010	Submission deadline for regular papers, tutorial and workshop proposals
8 January 2011	Submission of industrial presentation proposals
8 February 2011	Notification of acceptance to authors
8 March 2011	Camera-ready version of regular papers required
16 May 2011	Industrial presentations, tutorial and workshop material required
20-24 June 2011	Conference

Topics

Over the years Ada-Europe has established itself as an international forum for providers, practitioners and researchers into reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the development and maintenance of long-lived, high-quality software systems for a variety of established and novel application domains. The program will allow ample time for keynotes, Q&A sessions, panel discussions and social events. Participants will include practitioners and researchers representing industry, academia and government organizations active in the promotion and development of reliable software technologies.

All contributions, whether regular papers, industrial presentations, tutorials or workshops, should address the topics of interest to the conference, which for this edition include but are not limited to:

- **Methods and Techniques for Software Development and Maintenance**: Requirements Engineering, Object-Oriented Technologies, Model-driven Engineering, Formal Methods and Supporting Toolsets, Re-engineering and Reverse Engineering, Reuse, Software Management.
- **Software Architectures**: Architectural Styles, Service-Oriented Architectures, Cloud Service Model, Design Patterns, Frameworks, Architecture-Centered Development, Component and Class Libraries, Component-based Design and Development.
- **Enabling Technologies**: Software Development Environments, Compilers, Debuggers, Run-time Systems, Middleware Components, Concurrent and Distributed Programming, Ada Language and Technologies.
- **Software Quality**: Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems.
- **Theory and Practice of High-Integrity Systems**: Real-Time, Distribution, Fault Tolerance, Security, Reliability, Availability, Trust and Safety, Language Vulnerabilities.
- **Embedded Systems**: Multicore Architectures, HW/SW Co-Design, Reliability and Performance Analysis.
- **Mainstream and Emerging Applications**: Manufacturing, Robotics, Avionics, Space, Health Care, Transportation, Energy, Fun and Business Games, Telecommunication, etc.
- **Experience Reports**: Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics.
- **The Future of Ada**: Glimpses on the ongoing language revision as it reaches standardization; positioning in the market and in education; where should Ada stand in the software engineering curriculum; lessons learned on Ada Education and Training Activities with bearing on any of the conference topics.

In cooperation with ACM SIGAda (approval pending)

Call for Regular Papers

Authors of regular papers which are to undergo peer review for acceptance are invited to submit original contributions. Paper submissions shall be in English, complete and not exceeding 14 LNCS-style pages in length. Authors should submit their work via the Web submission system accessible from the Conference Home page. The format for submission is solely PDF. Should you have problems to comply with format and submission requirements, please contact the *Program Chairs*.

Proceedings

The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer, and will be available at the start of the conference. The authors of accepted regular papers shall prepare camera-ready submissions in full conformance with the LNCS style, not exceeding 14 pages and strictly by 8 March 2011. For format and style guidelines authors should refer to http://www.springer.de/comp/lncs/authors.html. Failure to comply and to register for the conference by that date will prevent the paper from appearing in the proceedings.

The conference is ranked class A in the CORE ranking and is listed among the top quarter of CiteSeerX Venue Impact Factor.

Awards

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

Call for Industrial Presentations

The conference also seeks industrial presentations which deliver value and insight, but may not fit the selection process for regular papers. Authors of industrial presentations are invited to submit a short overview (at least one page) of the proposed presentation by 8 January 2011. Please follow the submission instructions on the conference website. The *Industrial Committee* will review the proposals and make the selection. The authors of selected presentations shall prepare a final short abstract and submit it by 16 May 2011, aiming at a 20-minute talk. Accepted authors will also be invited to submit corresponding articles for publication in the Ada User Journal, which will host the proceedings of the Industrial Program of the Conference. For any further information please contact the *Industrial Chair* directly.

Call for Tutorials

Tutorials should address subjects that fall within the scope of the conference and may be proposed as either half- or full-day events, to be scheduled at either end of the conference week. Proposals should include a title, an abstract, a description of the topic, a detailed outline of the presentation, a description of the presenter's lecturing expertise in general and with the proposed topic in particular, the proposed duration (half day or full day), the intended level of the tutorial (introductory, intermediate, or advanced), the recommended audience experience and background, and a statement of the reasons for attending. Proposals should be submitted by e-mail to the *Tutorial Chair*. The authors of accepted full-day tutorials will receive a complimentary conference registration as well as a fee for every paying participant in excess of five; for half-day tutorials, these benefits will be accordingly halved. The Ada User Journal will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

Workshops on themes that fall within the conference scope may be proposed. Proposals may be submitted for half- or full-day events, to be scheduled at either end of the conference week. Workshop proposals should be submitted to the *Conference Chair*. The workshop organizer shall also commit to preparing proceedings for timely publication in the Ada User Journal.

Call for Exhibitors

The commercial exhibition will span the three days of the main conference. Vendors and providers of software products and services should contact the *Exhibition Chair* for information and for allowing suitable planning of the exhibition space and time.

Grants for Reduced Student Fees

A limited number of sponsored grants for reduced fees is expected to be available for students who would like to attend the conference or tutorials. Contact the *Conference Chair* for details.

Ada-Europe Launches
Annual Student Programming Contest *"The Ada Way"*

VALENCIA, Spain (June 16, 2010) – On the occasion of Ada-Europe 2010, the 15th annual Conference on Reliable Software Technologies, Ada-Europe, the international organization that promotes the knowledge and use of Ada in European academia, research and industry, launched an annual Student Programming Contest under the provisional title of "*The Ada Way".*

The contest will be a yearly competition among student teams, whereby each team must have a codename and a logo, a university affiliation, and the endorsement by an educator. The theme of this year's contest will be announced by September 1, and submissions will be accepted until April 30 of the following year. A *Steering Committee* composed of representatives of promoting institutions will oversee the organization of this contest.

Submissions will be marked by an *Evaluation Committee* composed of leading Ada experts, such as John Barnes (author of the famous *Programming in Ada* books), S. Tucker Taft (leader of the *Ada 95* language revision), Ed Schonberg (co-author of the open-source *GNAT Ada compiler and toolset*), Joyce Tokar (convenor of the *ISO working group* on the Ada language standards), etc.

The winning team will be notified by May 31. The Steering Committee will offer the winners one free registration, accommodation, and airfare to the *Ada-Europe conference*, a slot in the conference program, publication space in the *Ada User Journal*, and visibility in other media. Additional prizes might be offered.

Ada-Europe wants the competition to be fun and educational. The theme of the contest will be determined by the Steering Committee, and shall be intellectually challenging and elating. Evaluation criteria shall include correctness, clarity and readability of the code, ingenuity, cuteness, and time and space efficiency.

Submissions shall include the source code and the User Manual. The code must run out-of-the-box when following the User Manual's instructions. The implementation does not need to be 100% Ada, but of course the essence must be in Ada, and only the Ada code will be part of the evaluation. Tullio Vardanega, president of Ada-Europe, stated: *"The winning submission must be a reference for good Ada programming, software design, and innovation."*

About Ada-Europe
Ada-Europe is the international non-profit organization that promotes the knowledge and use of Ada into academia, research and industry in Europe. Current member organizations of Ada-Europe are: Ada-Belgium, Ada in Denmark, Ada-Deutschland, Ada-France, Ada-Spain, Ada in Sweden and Ada-Switzerland. Ada-Europe also includes and welcomes individual members from other European countries with no national organization, and has a total membership in the region of 300.

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

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

ISO/IEC JTC 1/SC 22/WG 9 N506

WG9 Letter to the Community: Maintenance and Revision of the Ada Programming Language

With Ada 2012 coming to fruition, it is appropriate to remind the community that suggestions for future enhancements to the language are always welcome. Remember that two critical organizations are involved, namely WG 9 and the ARG, and both of them encourage participation in their activities by all who are interested in Ada.

ISO/IEC JTC 1/SC 22/WG 9 is responsible for the maintenance and revision of the Ada Programming Language and associated standards and technical report. The Ada Rapporteur Group (ARG) is charged by the WG 9 committee to maintain the Ada Reference Manual and to supervise the evolution of the language. The ARG is composed of programming language experts, implementers, and users of the language. The ARG receives input from the Ada community at large, in the form of Ada comments and Ada Issues, following a procedure established two decades ago. ISO rules require that a standard be revised periodically, and the ARG is currently engaged in the preparation of an Amendment to Ada 2005, that will define the next version of the language. WG 9 has given the next edition of the Ada language the working name Ada 2012, which sets narrow boundaries for the revision work. The reasons for this choice and their impact on ARG work are outlined below.

The process of language evolution, from Ada 83 to Ada 95 to Ada 2005, and on to Ada 2012, has itself changed over time. The resources that were invested in the Ada9X process, which led to the Ada 95 standard, were simply not available for the next revision, and are not available today, The WG 9 committee, after discussions with the ARG and with members of the Ada community, has instructed the ARG to complete the Amendment to Ada 2005 so that ISO standardization of the new version can be completed by 2012. This is a relatively short horizon, but it matches the interval between previous releases, demonstrates that the language continues to evolve, and at the same time guarantees that the changes to the language are evolutionary and do not present an undue implementation burden on existing compilers.

The ARG welcomes suggestions for language enhancements at any time. These should be sent to *ada-comment@ada-auth.org*. If the suggestion is for a correction to an error in the existing Reference Manual and the error is significant, it will certainly be included in the Amendment. If it is a large-scale enhancement it may be worth discussing for the next version of the language (circa 2017). If it is a suggestion for new libraries, the most productive approach will be for interested members of the community to implement the libraries and put them in circulation before eventual standardization, in order to receive feedback from users. The guidelines presented for the Ada 2005 revision (see *http://archive.adaic.com/news/pressrelease/call4apis.html*) are still active.

The software industry has changed radically in the last decade, and software evolves much more rapidly than language standards, through informal mechanisms that are a universe away from the staid ISO procedures. For those interested in more ambitious extensions to the language, it is worth recalling that there is an open-source Free Software implementation of Ada 2005 that anyone can use as a workbench for language design.

The ARG has focused its work on two areas of particular interest to the Ada community: improved facilities for program correctness, and enhanced container libraries. There are numerous other proposed enhancements, and the interested reader can find the current state of these at *http://www.ada-auth.org/AI05-SUMMARY.HTML*. Some of these proposals originated with members of the ARG, and some others from members of the community at large. We consider that the basic ingredients of the Amendment are already at hand, even though much work remains to determine which of the Ada Issues already under discussion will in fact be incorporated in the language.

We are confident that the ongoing work of the ARG, with the received input from all interested parties, will converge to a design that satisfies the needs of Ada programmers, and remains faithful to the spirit and "feel" of the language.

The ARG and WG 9 encourages members of the Ada community at large to use the guidelines outlined above to provide input to WG 9 and ARG for needed revisions and upgrades to the Ada programming language.

Experience in programming device drivers with the Ravenscar profile

Jorge López, Ángel Esquinas, Juan Zamorano, Juan Antonio de la Puente
Universidad Politécnica de Madrid, ETSIT UPM, E 28040 Madrid, Spain

Abstract

The Ravenscar profile defines a subset of Ada tasking that can be statically analysable for real-time properties. The implications of the Ravenscar profile and other commonly used high-integrity restrictions for developing device drivers are analysed in the paper, and some guidelines are provided based on the analysis. The technical content of the paper is based on the authors' experience in developing communication drivers for the Open Ravenscar real-time Kernel (ORK) that are well suited for space on-board applications. A reference architecture for device drivers is proposed, and two instances of drivers based on it are described.

Keywords: Ada 2005, real-time systems, Ravenscar profile, device drivers, low-level programming.

1 Introduction

The Ravenscar profile [5] defines a subset of Ada tasking that can be used to develop real-time systems with predictable, analysable temporal behaviour. It is aimed at high-integrity applications that can eventually undergo a certification process with respect to some domain-specific standard. The profile has been widely accepted in academy and industry, and a number of industrial-grade implementations are available which can be used to develop highly critical systems. All of them include cross-compilation chain and a runtime system supporting the static tasking model defined by the profile.

A Ravenscar runtime system typically includes a tasking kernel, as well as some basic device drivers, e.g. for one or more system clocks and a serial line for debugging purposes. However, embedded real-time systems usually include specific hardware devices for which appropriate drivers have to be developed, often as part of an application development process. Driver programming requires accessing device controller registers, even at the bit level, and synchronizing the I/O operations with the CPU, usually by means of interrupts [7]. The Ada language [12] includes a number of low-level constructs for this purpose, among which the main ones are:

- *Representation clauses* can be used to represent hardware registers, including bit-wise structures and the addresses where they are located, by means of data types, data objects, and type and object attributes.
- *Protected procedures* can be used as interrupt handlers. The enclosing protected objects may include data objects and other protected operations that can be used by the application tasks to interact with the device.

Other useful low-level elements of the Ada language include storage address handling, machine code insertions, and shared variable control pragmas.

The Ravenscar profile explicitly allows most of the above features, provided they are used in such a way that the structure of the program remains static (e.g. dynamic handler attachment is forbidden). However, the profile excludes some useful elements that are part of common programming patterns for drivers, such as multiple entries in protected entries and requeue statements (see e.g. [5]). Furthermore, since the profile only addresses the tasking aspects of the language, additional restrictions are usually set on sequential constructs in order to enforce temporal predictability and support different kinds of static analysis in high-integrity applications [14]. Such restrictions may limit the use of some elements that are commonly used in device drivers (e.g. access types). Therefore, developing device drivers for high-integrity real-time systems may require additional effort from the programmer in order to overcome the restrictions in the expressive power of the language that are imposed in order to comply with the predictability and reliability properties required from such systems.

In the next section the overall difficulties in developing device drivers for high-integrity systems are analysed, and some general guidelines are proposed. Section 3 describes the authors' experience in developing drivers for a family of on-board embedded computers in the space domain, and a software architecture for device drivers is proposed. Two instances of communication drivers derived from this architecture are described in section 4. Finally, conclusions of the work and future work plans are explained in section 5.

2 Device drivers and high-integrity restrictions

2.1 Ravenscar restrictions

The Ravenscar profile is defined by three pragmas and a set of restrictions [12]. The pragmas specify the dispatching and locking policies to be FIFO_Within_Priorities and Ceiling_Locking, respectively, and require potentially blocking operations within protected operations to be detected. The restrictions define a static, analysable tasking model [13].

The profile explicitly allows protected procedure interrupt handlers to be declared using pragma Attach_Handler, forbidding only the use of the dynamic attachment features defined in the Ada.Interrupts package. Therefore, the basic language elements for programming device drivers are available in Ravenscar programs, and most of the Ravenscar restrictions raise no problems in this respect.

However, some common programming patterns for drivers, such as the simple two-step pattern shown in listing 1 are not allowed by the profile. This pattern uses two features forbidden by the profile, multiple protected entries and requeue, to perform an input-output operation in two steps: first Start_IO is called to setup the device registers as needed. Then the call is requeued to End_IO, awaiting the completion of the operation to be signalled by an interrupt. When the interrupt arrives, the handler opens the End_IO barrier and the operation completes. Notice that End_IO is private as it is only invoked by the requeue statement in Start_IO and thus cannot be called by other program units.

This pattern can easily be transformed into one that does not make use of requeue, as shown in listing 2, provided that the driver is only used by one application task. In this case the task can call Start_IO as a procedure and consequently exit the protected object. A second explicit call to entry End_IO has to be made in order to await the arrival of the interrupt, which is handled as before.

Notice that allowing only one application task to use a device driver is quite natural in Ravenscar programs. Otherwise, two tasks might be queuing on the protected entry of the drive, which is forbidden by the profile.

2.2 Other high-integrity restrictions

In addition to the Ravenscar tasking restrictions, restrictions on the sequential part of the language are often enforced on high-integrity systems, in order to enhance their robustness and predictability and enable advanced verification techniques to be used [14]. Some common restrictions are:

- No_Allocators
- No_Unchecked_Access
- No_Dispatch
- No_Recursion
- No_IO
- No_Exceptions
- No_Access_Subprograms

Most of the restrictions in the above list do not raise any special problem for programming drivers. The last two ones, however, deserve some further attention. No_Exceptions prevents exceptions to be used to handle

Listing 1 Two-step driver

```
protected Driver is
 entry Start_IO;
private
 entry End_IO;
 procedure Handler;
 pragma Attach_Handler(Int_ID, Handler);
 Ready : Boolean := True;
 Finished : Boolean := False;
end Driver;
protected body Driver is
  entry Start_IO when Ready is
  begin
    ...
    Ready := False; Finished := False;
    requeue Complete_IO;
  end Start_IO;
 entry End_IO when Finished is
 begin
   ...
   Ready := True;
 end End_IO;
 procedure Handler is
 begin
   ...
   Finished := True;
  end Handler;
end Driver;
```

Listing 2 Ravenscar-compliant two-step driver

```
protected Driver is
 procedure Start_IO;
 entry End_IO;
private
 procedure Handler;
 pragma Attach_Handler(Int_ID, Handler);
 Finished : Boolean := False;
end Driver;
protected body Driver is
  procedure Start_IO is
  begin
    ...
    Finished := False;
  end Start_IO;
 entry End_IO when Finished is
 begin
   ...
 end End_IO;
 procedure Handler is
 begin
   ...
   Finished := True;
  end Handler;
end Driver;
```

hardware errors in I/O operations. Since error detection and signalling is a key element of most device drivers, lower-level mechanisms such as error status variables or error parameters in subprograms must be used. Some implementations (see e.g. [1]) allow a fine-grain control of exceptions by specifically restricting the use of exception handlers or exception propagation, but the results are roughly the same. As for No_Access_Subprograms, its implications are not obvious for simple drivers, but this restriction may cause problems for drivers with initialization-time configuration, as discussed in section 4.

3 A generic driver architecture

Computers are built up with a set of modules of three basic types: processors, memories and I/O devices, the latter being in charge of communicating with the computer environment through the so-called peripheral devices. Nowadays, the common way to interconnect computer components is by means of a computer bus or, more often, a computer bus hierarchy. The usual arrangement is to interconnect components on the same board with a local bus, and communicate different boards by means of a backplane bus.

The I/O device interface from the processor side usually consists of several registers that can be classified as:

- **Status registers**, which are used to store the status of the attached device. The processor can check the status of a device by reading its status registers.
- **Control registers**, which accept commands from the processor that are decoded by the I/O module in order to issue the corresponding request to the peripheral device.
- **Data registers**, which perform data buffering in order to decouple the different transfer rates of the main memory and the peripheral device.

A device driver is a software module that provides application code with access to a peripheral device. The application code invokes driver operations in order to interact with the device by means of commands that are sent to the device. When the device sends data or control information back to the driver, it completes the operation at the application level by returning from the call or by invoking other routines in the application. Device drivers also provide for interrupt handling and other synchronization operations. Due to their strong interaction with the device, device drivers are hardware-dependent and operating systems-specific in nature.

In order to enable the driver to interact with the device, the I/O registers must be allocated a unique address in an address space that can be made accessible to the processor. For port-based addressing architectures, the I/O registers can be accessed by subprograms including assembly code instead.

In a similar way, interrupt and DMA (Direct Memory Access) request lines have to be properly set and identified. Real-time modular computers are commonly based on standard backplane buses, such as VME, EISA or PCI, where processors, memory modules, and I/O boards are plugged. Some backplane buses provide jumpers or micro-switches for manual configuration of the addresses in each board. Other modular buses do not provide such low-level mechanisms, and the board configuration is done by reading board parameters and writing the settings on board registers, using a separate configuration address space. In this case, an initialization routine has to be developed as part of the device driver, which is commonly called a “plug and play” routine. Such a routine typically includes locating I/O devices by exploring boards that are connected to the system, calculating proper settings for the device, and writing them onto the configuration registers.

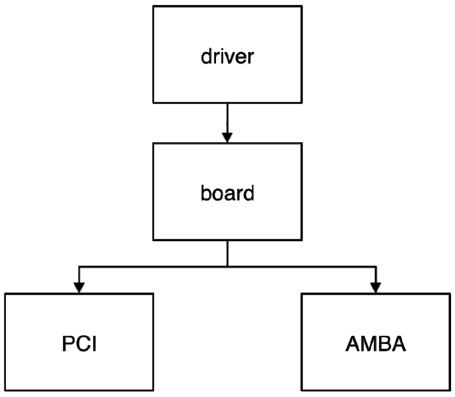

Figure 1 Generic driver architecture

Once initialized, the main functions of a device driver are translating higher-level commands into device-specific commands, reading device states, synchronizing the operation of the device with the processors, and transferring data transfer to or from main memory.

The software architecture should ideally reflect this organization, by providing separate components for handling peripheral devices and buses. Figure 1 shows a generic architecture for a device driver that uses an AMBA bus [2] as a local bus, and a PCI bus [18] as a backplane bus. It is modelled after other software architectures that have been successfully implemented for other devices [4], [15], [17].

4 Communications drivers for ORK+ and LEON computers

4.1 Introduction

Two communication drivers for a particular platform used in space systems are described in this section to illustrate some concepts that must be taken into account for developing device drivers for high-integrity systems. The hardware platform is based on a LEON2 computer [3], a radiation-hardened implementation of the SPARC V8 architecture [19]. The software platform is based on the GNATforLEON compiler[1] and ORK+, the current version of the Open Ravenscar real-time Kernel [8][20].

[1] www.adacore.com

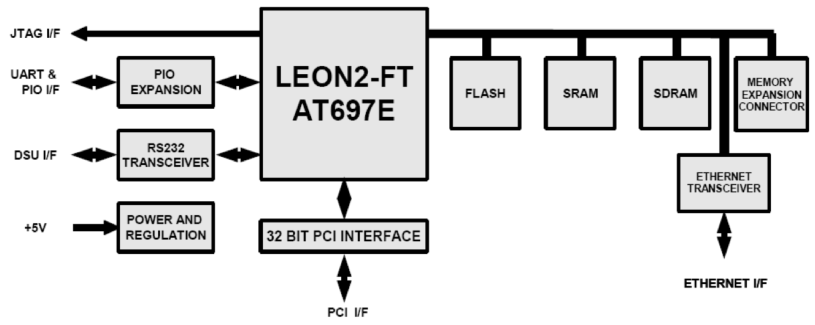

Figure 2 GR-CPCI-AT697 CPU board block diagram (reproduced from [10]).

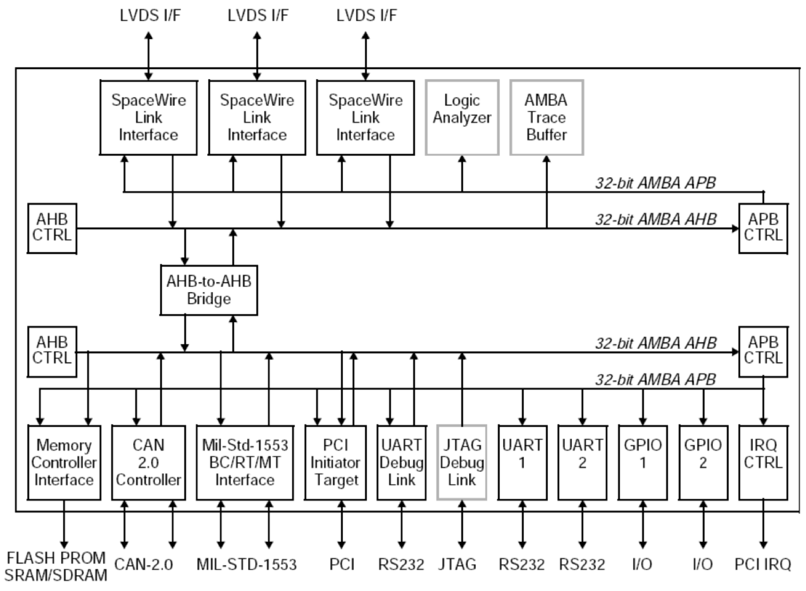

Figure 3 RASTA interface board block diagram (reproduced from [11]).

4.2 Hardware platform

GR-RASTA is a development and evaluation platform for LEON2 and LEON3-based spacecraft avionics built on a Compact PCI (cPCI) backplane bus. The computer has two cPCI boards:

- *GR-CPCI-AT697*: this is the processor board. It includes a LEON2 processor and memory. Its structure is shown in figure 2. The board has a PCI bridge to access the cPCI backplane bus.
- *GR-CPCI-XC4V*: this is an interface board based on a FPGA which has several I/O modules, including three SpaceWire links. Its design is based on an AMBA[2] bus to which the units are connected. It also has a PCI bridge to access the cPCI backplane bus. The structure of this board is shown in figure 3.

System software is usually unaware of the bus hierarchy, aside from the startup configuration of the plug-and-play feature. However, it is important to take into account the "endianness" of the different buses of the hierarchy as it has a strong influence on the definition of device registers. The SPARC v8 architecture, and therefore LEON, is big-endian. This is also the byte ordering of the AMBA buses in LEON processors. However, the PCI bus is little-endian, as it was mainly developed for Intel x86 processors. In this way, I/O device multibyte registers will suffer byte twisting as shown in figure 4. This issue must be taken into account for PCI I/O device multibyte registers as well as for DMA transfers. Accordingly, PCI hosts and PCI DMA I/O devices must be properly initialized.

[2] The Advanced Microcontroller Bus Architecture (AMBA) is a system and peripheral bus widely used in System-on-a-chip (SoC) designs.

4.2 The SpaceWire Device

The Gaisler SpaceWire (GRSPW) core handles the lower-level layers of the SpaceWire protocol [9]. It is an intelligent I/O device with Direct Memory Access (DMA) and interrupt-based synchronization with the CPU. The interrupt service routine (ISR) is expected to read the status registers so as to check if the operation has been successfully completed.

The GRSPW core has three main parts:

- The link interface, which handles the communication on the SpaceWire network and consists of a transmitter, receiver, and FIFO interfaces. FIFO interfaces are provided to the DMA engines and are

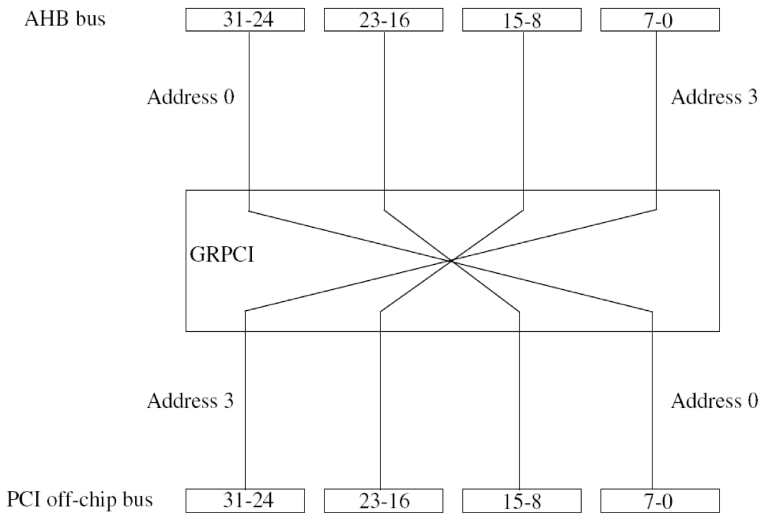

Figure 4 AMBA to PCI bus byte twisting.

used to transfer a number of characters (N-Chars in the following) between the AMBA and SpaceWire domains during reception and transmission.

N-Chars are sent when they are available from the transmitter FIFO and there are credits available. The credit counter is automatically increased when flow control tokens (FCT) are received and decreased when N-Chars are transmitted. Received N-Chars are stored to the receiver N-Char FIFO for further handling by the DMA interface.

- The AMBA interface, which consists of the receiver and transmitter DMA engines.

  The receiver DMA engine reads N-Chars from the N-Char FIFO and stores them on a DMA channel. Reception is based on descriptors located in a consecutive area in memory that hold pointers to buffers where packets should be stored. When a packet arrives it reads a descriptor from memory and stores the packet to the memory area pointed by the descriptor.

  Before reception can take place, a few registers need to be initialized, such as the node address register, which needs to be set to hold the address of this SpaceWire node. The link interface has to be put in the run state before any data can be sent. Also, the descriptor table and control register must be initialized.

  The transmitter DMA engine reads data from the AMBA bus and stores them in the transmitter FIFO for transmission on the SpaceWire network.

- The RMAP handler is an optional part of the GRSPW and handles incoming packets which are determined to be RMAP (Remote Memory Access Protocol) commands.

4.3 SpaceWire Driver Architecture

Figure 5 contains a diagram of the software organization of the GRSPW driver, which is an instance of the generic architecture described previously (see figure 1). The driver has four main components:

- The PCI driver component, which provides data type definitions and operations for reading and writing the PCI configuration registers.
- The AMBA driver component, which provides data type definitions and operations for scanning the AMBA configuration records.
- The RastaBoard driver component, which provides a common interface for drivers using the GR-RASTA board, as well as hooks for interrupt handlers to be called upon reception of the single hardware interrupt issued by the board.

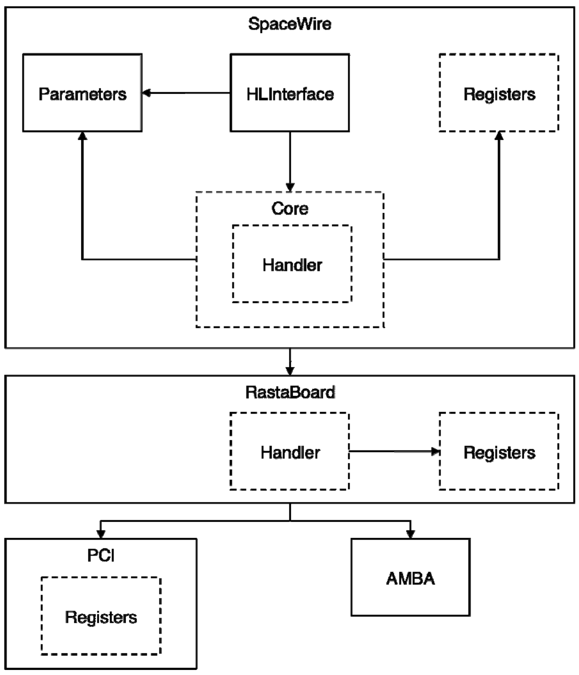

Figure 5 SpaceWire driver architecture.

- The SpaceWire driver component, which provides all the software items required by application programs to initialize and use the SpaceWire cores included in the GR-RASTA computer platform.

The components of the SpaceWire driver are:

- HLInterface: contains the higher-level interface for application programs, consisting of type definitions and operations initializing the SpaceWire devices, setting their node addresses, and sending and receiving data packets.
- Parameters: contains the definitions of all the parameters that can be configured by the application programmer.
- Core: contains all the code that interacts with the device registers in order to implement the I/O operations.

  This component exports a set of interface operations, which are used to implement the HLInterface operations. The component implements all the device operations in terms of the device registers and other hardware characteristics.
- Handler contains the device interrupt handler, which is invoked on the completion of I/O operations. There is a single interrupt for all the three SpaceWire devices, and a synchronization object for each of the transmit and receive sections of each SpaceWire hardware device.[3] Each occurrence of the interrupt is signalled to the appropriate synchronization object by identifying the device and function that has caused the interrupt.
- Registers: contains register and bit field definitions, as well as other data definitions that may be required to interact with the device.

4.4 Serial driver

There are two UARTs (Universal Asynchronous Receiver-Transmitter) in the RASTA board that provide an interface between an APB bus and a RS-232 serial line. Each UART provides the functionality for asynchronous serial communications, supporting data frames with 8 data bits, one optional parity bit, and one stop bit. As usual, it is also possible to configure the baud rate and the flow control.

UART devices are not message-oriented as SpaceWire devices, but character-oriented devices, i.e. an UART I/O operation involves just one character. Nevertheless, higher level software usually needs to send or receive a set of characters which builds up a message. In order to provide this functionality, the driver includes two separate memory buffers for storing messages:

- Transmit buffer: when higher level software sends a message, the corresponding data are pushed into this buffer and then the transmission starts until the buffer is empty. The interrupt service routine is in charge of transferring data from the buffer to the transmitter register.
- Receive buffer: when a data item is received, the interrupt service routine transfers it from the receiver register to this buffer. In this way, higher level software can receive messages by getting the data from this buffer.

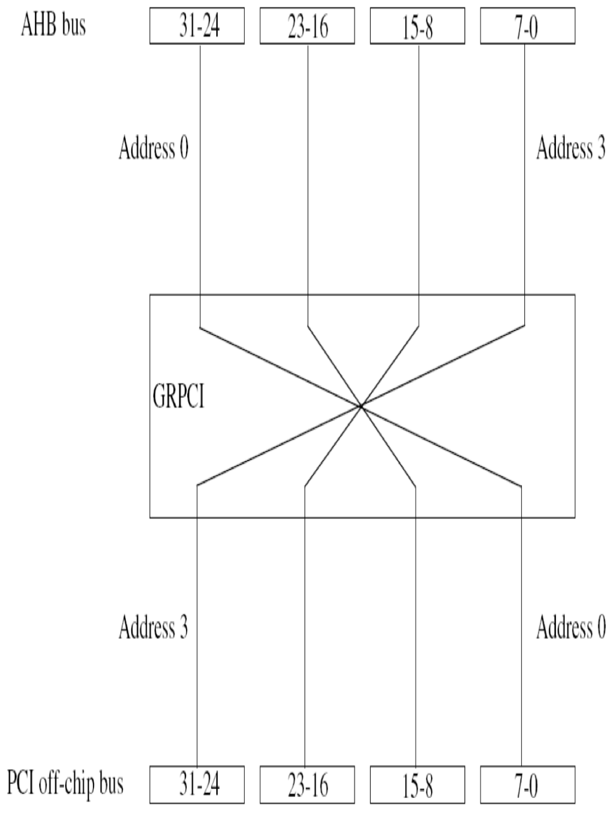

Figure 6 UART buffers arrangement.

These intermediate buffers are stored in main memory, and their sizes can be specified with the Buffer_Size parameters (declared in Uart.Parameters). If the value of these parameters is changed, the driver needs to be recompiled. Figure 6 shows the data flow between the UART registers and the intermediate buffers.

It must be noticed that the two-step driver pattern would have been useful when calling the driver's high level operations HL.Write and HL.Read. Both operations deal with messages and they try to read or write a set of characters from or to intermediate buffers. A call to HL.Read can be made with a message length greater than the currently stored in the receive buffer and thus the calling task may have to wait for the arrival of the rest of the message. As shown in section 2, this can be done in full Ada by using a requeue statement, but it has to be must be transformed into the alternate pattern shown in listing 2 in order to comply with the Ravenscar profile.

Figure 7 contains a diagram of the software architecture of the GRUART driver, which is an instance of the generic architecture described in section 3.

[3] The GR-RASTA interface board has three SpaceWire devices.

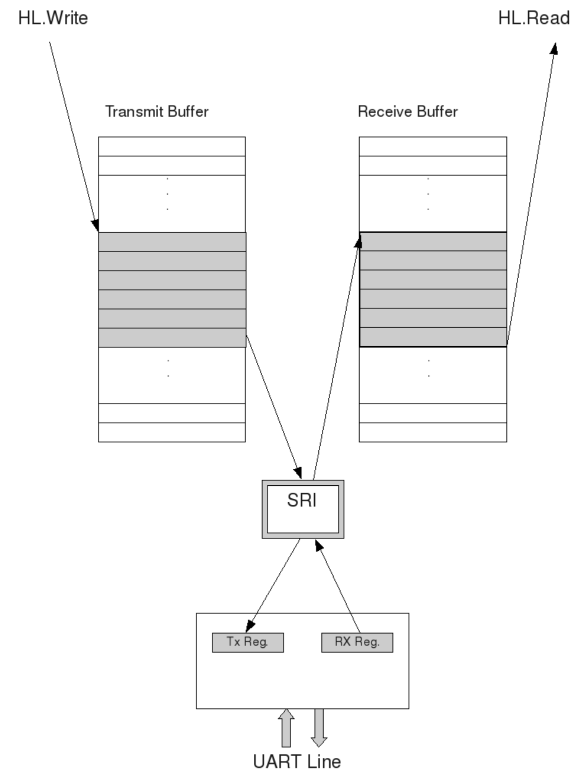

Figure 7 UART driver architecture.

5 Conclusions

The main issues related to writing device drives in Ravenscar Ada have been examined in the paper. A first conclusion is that the low-level mechanisms of the Ada language make it comparatively simple to develop device drivers in a high-level language. Features such as representation clauses and protected interrupt handlers allow the designer to build high-level abstractions of the hardware and greatly simplify writing the functional code of the drivers. Using record fields to name register bit groups improves the code readability compared to the lower-level bit mask approach used by other languages.

The good news is that these useful mechanisms are compatible with the Ravenscar profile, and thus can be used to build device drivers for high-integrity embedded real-time systems. The only potential problem that has been identified is the inability to use the requeue statement to write interrupt drivers using the well-known two-step synchronization pattern. However, a simple workaround has been proposed that only requires the restriction that a protected entry can only be called by one task.

A software architecture that can be used to develop device drivers for LEON computers has been introduced in the paper. Two driver instances for communication devices have been built based on the architecture. The authors' experience has been very positive, and is currently being continued with the developing of additional device drivers for the ORK+ real-time kernel and the GR-RASTA LEON computer boards within ESTEC, the European Space Research and Technology Centre of ESA.

Acknowledgments

This work has been funded in part by the Spanish Ministry of Science, project TIN2008-06766-C03-01 (RT-MODEL), and by the European Space Agency, ESTEC/Contract No. 21392/08/NL/JK.

References

[1] AdaCore (2009). *GNAT Reference Manual.*

[2] ARM (2003). *AMBA 3.0 Specification*.

[3] Atmel (2005). *Rad-Hard 32 bit SPARC V8 Processor* —AT697E.

[4] D. Berjón (2005). *Desarrollo de un subsistema fiable de comunicación para sistemas de tiempo real*. Master's thesis, ETSIT- UPM. In Spanish.

[5] A. Burns, B. Dobbing and G. Romanski (1998). *The Ravenscar tasking profile for high integrity real-time programs*. In L. Asplund (ed) Reliable Software Technologies—Ada-Europe'98. LNCS 1411, Springer-Verlag, pp 236-275.

[6] A. Burns and A. Wellings (2007). *Concurrent and Real-Time Programming in Ada*. Cambridge University Press.

[7] A. Burns and A. Wellings (2009). *Real-Time Systems and Programming Languages*. 4th edn. Addison-Wesley.

[8] J.A. de la Puente, J.F. Ruiz and J. Zamorano (2000). *An open Ravenscar real-time kernel for GNAT*. In H.B. Keller and E. Plödereder (eds), *Reliable Software Technologies—Ada-Europe 2000*. LNCS 1845, Springer-Verlag, pp 5–15.

[9] ECSS (2008). ECSS-E-ST-50-12C: *Space engineering — SpaceWire — Links, nodes, routers and networks*.

[10] Gaisler Research (2005). *LEON2 Processor User's Manual*.

[11] Gaisler Research (2006). *RASTA Interface Board* FPGA User's Manual.

[12] ISO/IEC: Std. 8652:1995/Amd 1:2007. *Ada 2005 Reference Manual. Language and Standard Libraries.* LNCS 4348, Springer-Verlag.

[13] ISO/IEC: TR 24718:2005. *Guide for the use of the Ada Ravenscar Profile in high integrity systems*. Based on the University of York Technical Report YCS-2003-348 (2003).

[14] ISO/IEC: TR 15942:2000. *Guide for the use of the Ada programming language in high integrity systems*.

[15] D.S. Morilla (1995). *Programación en Ada del LANCE Am7990*. Master's thesis, FI-UPM. In Spanish.

[16] PRAXIS Ltd (2008) *The SPARK Ravenscar Profile*.

[17] J.E. Salazar, J.E.: *Desarrollo de un driver para un sistema espacial de alta integridad*. Master's thesis, FI- UPM. In Spanish.

[18] T. Shanley and D. Anderson (1999). *PCI System Architecture*. 4th edn. Mindshare Inc.

[19] SPARC International (1992). *The SPARC architecture manual: Version 8.* Prentice-Hall.

[20] S. Urueña, J.A. Pulido, J. Redondo and J. Zamorano (2007). *Implementing the new Ada 2005 real-time features on a bare board kernel.* Ada Letters, vol XXVII no 2, p 61–66.

Executable Requirements in a Safety-Critical Context with Ada

Christophe Baillon, Shanti Bouchez-Mongardé

SOGILIS, 10 ter boulevard Gambetta, 38000 Grenoble, France, http://sogilis.com;
email: { cba , shanti }@ sogilis.com

Abstract

When people who need the software and people who build the software do not understand each other, the success of a project may be impacted. In the same way, when it is difficult to know which test case corresponds to a given requirement, or if each and every requirement is fully covered by the test suite, we have a traceability issue that may also impact the success of the project.

During the last few years, the agile community has suggested a new development paradigm in order to address this traceability issue. This approach is called Behaviour Driven Development (BDD). It is based on a new way of expressing requirements using a common language understandable by all parties at stake.

After having described the Behaviour Driven Development in details, we introduce the XReq tool, an Open Source project developed by SOGILIS and part of the Open-DO project. It is designed to bring the Behaviour Driven Development to the Ada language and other statically typed languages. It also aims at facilitating the traceability of High and Low Level Tests in the context of DO-178B projects.

Keywords: Behaviour Driven Development, Ada, Requirements, Testing, Safety-Critical Projects, DO-178B.

1 Introduction

The main agile principles have been formulated in the Agile Manifesto . The core ideas behind these principles are:

- Be able to deliver often;
- Never disregard quality;
- Reduce the cost of changes;
- Eliminate ambiguities between the customer and the developers.

A very common problem in every software project is keeping the documentation up-to-date at all time.

Recently, a trend has been emerging in software development, originally from the Agile community: **Prefer executable requirements over static Documentation.** Instead of having well-known static documents that expose requirements and specifications for which we never know if the corresponding test cases are up-to-date, the idea is to tie together requirements and test cases with the help of a well defined language .

The expression of such a language is quite difficult though. At the very beginning, some attempts have been made using syntactic sugar with languages like Java, C# or Python in order to express requirements in a way that makes them understandable by the customer or functional engineers who may not be fluent with programming languages. As we can imagine, this choice was not very satisfactory because a programming language is not well suited to express a business need while hiding implementation details.

Lately, the Ruby community has found an interesting solution with the Cucumber project[1]. Cucumber understands Gherkin[2], a business readable Domain Specific Language (DSL) that lets you describe your software's behaviour without having to reveal any detail on the actual behaviour's implementation.

By expressing the requirements in both a readable and executable manner, they become both documentation and automated tests. Every requirement is then directly mapped to the code. When all the tests are run successfully, we are therefore guaranteed that all our requirements have been covered. Executable requirements are defined in such a way that they become automated acceptance tests[3] themselves. XReq is a project which allows requirements to be written just like with Cucumber, but for safety-critical projects using the Ada 2005 language .

This is especially well suited for DO-178B projects that need good traceability between requirements and tests for the High Level Requirements (HLR) and High Level Tests (HLT). XReq can also be used with Low Level Requirements (LLR) and Low Level Tests (LLT).

In the next sections, we explain the history of BDD, the XReq mechanism, how we write requirements, and how XReq maps these requirements to the Ada code. Here we talk about XReq for DO, a version fitted to the DO-178B world.

[1] http://cukes.info
[2] http://wiki.github.com/aslakhellesoy/cucumber/gherkin
[3] http://en.wikipedia.org/wiki/Acceptance_testing

1.1 Test Driven Development

Test Driven Development (TDD) is a software development practice that involves writing tests before writing the code being tested.

You start by writing a very small test for code that does not yet exist. At this point, you run the test and check that it fails. Only then you can start writing the code, actually just enough code to make that test pass, and no more. But TDD is not a testing practice at all. Instead, the goal of TDD is specification and not validation. It is a way to think about functionalities of your application before you write your business code. Of course, it is also a programming technique, but tests are here mainly for documentation purpose: each test explains how to use a module by providing examples. In doing so, we always have a good coverage of the code, and regressions are detected very early. The typical TDD cycle is:

- Write the test and imagine how you need to use the module being tested;
- Run the test, it should fail;
- Write the code to make the test pass, as simple as possible;
- When the test passes, refactor the code, and make sure every test passes.

TDD is appropriate for implementing design specifications but has showed some limitations for expressing high level requirements because programming languages are not appropriate to perform this task.

1.2 Behaviour Driven Development

Let's now go one step further. Behaviour Driven Development was born because many developers were having a hard time relating to TDD as a design technique.

In 2002 Ward Cunningham invented Fit[4]. It uses HTML tables to present the business people's examples, which are connected to test fixtures written by programmers. FitNesse[5] is based on Fit and adds a wiki approach to automated acceptance testing.

Around the same time, Dan North grew dissatisfied with TDD: “The deeper I got into TDD, the more I felt that my own journey had been [...] a series of blind alleys”, he said[6]. It started out as a reflection to get rid of the unit testing vocabulary (tests, assertions), and use the vocabulary of behaviour instead (e.g. a method should behave like this) and focus on making sensible naming schemes so that failing tests would be easier to fix, edit or delete if needed.

In late 2003, he started working on JBehave[7], which would later focus more on expressing requirements and acceptance criteria, mapping textual scenarios to executable bits of code in order to check the software behaves the way it should.

BDD is about writing context for our scenarios first: the user's role, what he wants the system to do for him, and what he would gain from this feature. Then you write a textual scenario of what exactly the user will do. Finally, you start a cycle that includes TDD itself: each line of the scenario is mapped to test code. We start the acceptance test and every time we have to use an object, a class, or a method, we write a unit test for this tiny bit of feature, then just enough code to make it pass, refactor, then go back to the acceptance test's next line. That way every single line of code in your project is there to fulfil the stakeholder's needs.

Every developer who starts practising TDD encounters the same issues:

1. What do I test first?
2. How far should I test?
3. What needs to be tested, what does not?

BDD answers these questions: a unit test is written only when an acceptance feature calls an object or method that did not exist already or did not behave the way it should. When the unit test passes, we launch the acceptance test again and move to the next failing step.

Examples will be provided in the following sections.

1.3 BDD Used With Legacy Software

Legacy software may be defined informally as software that were developed years ago and we don't know what to do with, but that is still performing a useful job. They are often critical to the operation of companies. These programs have been maintained for many years by hundreds of programmers, and while many changes have been made to the software, the supporting documentation may not be up-to-date or even missing. These factors contribute to the staggering cost of maintaining legacy systems. Consequently, there is an urgent need to find ways to make these programs more maintainable without disrupting the operation of the company. Michael C. Feathers even wrote: “To me, legacy code is simply code without tests” .

Most of the time, legacy software are out of control, and nobody is able to make them evolve while being sure that no regression is introduced. Using BDD and executable requirements with untested legacy software is a good way to build a step-by-step understanding of its behaviour.

2 Presentation of XReq

XReq has been launched to bring to the Ada world and in particular to the safety-critical area the missing tool for writing executable requirements. Beyond Ada, XReq also targets other compiled languages such as C and C++. In its design, it is strongly related to Cucumber as it is currently the most advanced tool for writing executable specifications.

[4] Framework for Integrated Test, available at http://fit.c2.com
[5] http://www.fitnesse.org
[6] http://dannorth.net/introducing-bd
[7] http://jbehave.org

XReq and Cucumber can understand the same file format for the specifications, but the behaviour of XReq had to be adapted to fit strong and statically typed languages such as Ada whereas Cucumber works with Ruby, a very dynamic and flexible language.

XReq works by translating the requirement expressed in natural language to Ada packages designed to test what is expressed in the requirements. We have the following pipeline:

1. Compile the requirements to test packages
2. Compile the test packages
3. Run the executable and see if the tests are successful.

The second step can be done automatically by XReq, but the user might want to do it manually in order to include specific libraries that might be needed for the test.

In order to compile the requirements, step definitions must be provided. Step definitions are regular Ada packages containing procedures that can be accessed in the requirement file in natural language. The correspondence between the sentences and the procedures is done using regular expressions provided in the package comments.

2.1 Requirements

Every requirement is gathered in a .requirement file. This file must begin with the keyword **Requirement:** followed by free text specifying the requirement. This text is not meaningful for XReq but serves to specify as clearly as possible the requirement in itself. The file is generally indented to clearly show the structure but it is optional.

2.2 Test Cases

The requirement includes several test cases intended to test the specific feature specified in the requirement. A test case must start with the keyword **Test Case:** followed by a simple description of the test.

Each test case consists of many different steps starting with one of the keywords **Given**, **When**, **Then** or **And**. The rest of the line describes in natural language what must be done at this stage of the test case.

If the test cases all share the same initialization steps, they can be gathered in a special section introduced by the keyword **Background:**.

The requirement can also contain test case templates that are quite similar to test cases except that they are used to create many test cases sharing the same structure.

2.3 Steps

The steps are phrases expressed in natural language that compose a part of the test case. When the test case is executed, the steps are executed in order.

The steps can begin with the following keywords:

Given	The purpose of Given is to put the system in a known state before the user or an external tool starts interacting with it (in the When steps). We should never talk about user interaction in Given steps.
When	The purpose of When steps is to describe the key action the user or the system performs (state transition).
Then	The purpose of Then steps is to observe outcomes. The observations should be related to the business value/benefit in the feature description. The observations should also be on some kind of output.
And	Can be used as a link-word to repeat the previous Given, When or Then keyword.

2.4 Step Definitions

Step definitions are Ada packages located in a special directory step_definitions next to the requirement files. There is absolutely no restriction on what is possible in those package except that it usually depends on the XReq library:

```
with XReqLib.General;
use  XReqLib.General;
```

In order to introduce a definition for a given step, a comment including the special keyword **@given**, **@when** or **@then** and followed by a regular expression matching the step text is put in front of the procedure implementing the step. The procedure must have the following signature:

```
-- @given ^this step works$
procedure Given_this_step_works
        (Args : in out Arg_Type);
```

The procedure argument Args is a complex data structure defined in the XReq library containing matches for the regular expression and other additional parameters that can be included in the requirement files that we will introduce later.

3 Calculator Example

The easiest way to describe the use of XReq is to give a simple example. Let's imagine your customer wants a very simple calculator. We are going to specify the requirement in the file calculator.requirement:

```
Requirement: Calculator
 In order to do simple calculations
 I want to be able to do simple operations
 (add, subtract, multiply and divide)

 Test Case: Add two numbers
  When I add “3” and “5”
  Then the result should be “8”

 Test Case: Subtract two numbers
  When I subtract “3” to “5”
  Then the result should be “2”
```

Now that this requirement is expressed in a very understandable way to non technical people, the programmer has to implement the step definitions related to these test cases in step_definitions/calculator_steps.ads:

```
with XReqLib.General;
use  XReqLib.General;
package Calculator_Steps is
   -- @when ^I add "([0-9]+)" and "([0-9]+)"$
   procedure Make_add (Args : in out Arg_Type);
   -- @when ^I substract "([0-9]+)" to "([0-9]+)"$
   procedure Make_sub (Args : in out Arg_Type);
   -- @then ^the result should be "([0-9]+)"$
   procedure Test_result (Args : in out Arg_Type);
end Calculator_Steps;
```

Note that this file can be generated by XReq.

The implementation is very simply expressed in step_definitions/calculator_steps.adb:

```
package body Calculator_Steps is
   Result : Integer;
   procedure Make_add (Args : in out Arg_Type) is
   begin
      Result := Integer'Value (Args.Match (1)) +
            Integer'Value (Args.Match (2));
   end Make_add;
   procedure Make_sub (Args : in out Arg_Type) is
   begin
      Result := Integer'Value (Args.Match (2)) -
            Integer'Value (Args.Match (1));
   end Make_sub;
   procedure Test_result (Args : in out Arg_Type) is
   begin
      Assert (Result = Integer'Value(Args.Match(1)));
   end Test_result;
end Calculator_Steps;
```

Note that due to the simplicity of our example, we actually only test the built-in + and – operators in Ada, not our actual calculator. In a more complex example, the step definitions are going to depend on some parts of the program you want to test.

Additionally, you may notice that we have to convert the numbers from String to Integer. This is because XReq captures parts of the step text, as a String. It has no knowledge whatsoever of what it should represent to you. If the requirement is not written properly and the string is not an Integer, the test will fail during the conversion.

The simplest way to compile the requirement is to specify it on the XReq command line, like:

```
xreq calculator.requirement
```

The step definition will be read, and the resulting package will be created under the tests directory:

-tests/requirement_calculator.ads

-tests/requirement_calculator.adb

These test packages can be used stand alone but will require you to know some of the internals of the XReq library. A more convenient way to compile the requirement is by specifying a test suite to generate:

```
xreq -x test_suite calculator.requirement
```

This will generate:

-tests/requirement_calculator.ads

-tests/requirement_calculator.adb

-tests/test_suite.adb

-tests/test_suite.gpr

The test suite can then be compiled using gnatmake on the generated GPR project file. XReq can compile it directly if the -m option is specified.

The test suite is an Ada program designed to run the test packages and generate a report either on the command line or in an HTML file. If invoked directly, the report will be:

```
$ tests/test_suite

Requirement: Calculator
  In order to do simple calculations
  I want to be able to do simple operations
  (add, subtract, multiply and divide)

  Test Case: Add two numbers
    When I add "3" and "5"
    Then the result should be "8"

  Test Case: Subtract two numbers
    When I subtract "3" to "5"
    Then the result should be "2"

2 test cases (2 passed)
4 steps (4 passed)
Finished in 0s
```

This shows that the tests pass, if an error happens, the description of the exception that occurred would be displayed below the failing step (shown in red) and the following steps would be skipped (shown in blue). The steps that pass are shown in green.

The HTML format provides additional features such as a table of contents, line numbering and more convenient display of errors and debug information.

4 Advanced features

Finding why the test fails can sometimes be difficult. For this reason, XReq step definitions can advertise debug information that will be displayed in debug mode. This make it possible to trace the reason of an error that happened few steps before the error is actually detected.

It is possible to specify more extensive information along with a step in a requirement file. This extended information can either take the form of unformatted text kept verbatim, or the form of tabular data, intended to represent tables.

Test case templates can be used to create different test cases using the same steps with slightly different values. The values of the template are specified after the template in a table introduced by the **Examples:** keyword. The following example is strictly equivalent to our two test cases in our previous example:

```
Test Case Template: Operations on two numbers
 When I <operation> "<op1>" <link> "<op2>"
 Then the result should be "<result>"
 Examples:
  | operation   | op1   | link  | op2   | result  |
  | add         | 3     | and   | 5     | 8       |
  | subtract    | 3     | to    | 5     | 2       |
```

5 Conclusion

Executable requirements is a technique that allows technical and non-technical people to understand each other, and to keep perfect traceability between requirements and test cases. This approach can be mixed smoothly with well-known software engineering techniques, or with formal approaches. Indeed, we often hear people who are opposing formal proof and testing. But we must understand that executable requirements are a way to capture the needs of the customer, and to map them into the code, to ensure we are developing the right software. Formal proof can just ensure that the code is right according to the developers' understanding. The two approaches are actually complementary. We could for example imagine mixing the SPARK language and XReq to express the requirements.

More information on the XReq project can be found on the Open-DO forge[8], and on the SOGILIS Website[9].

References

[1] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas, *Manifesto for Agile Software Development*, 2001, http: //agilemanifesto.org

[2] Jonathan S. Ostroff, David Makalsky and Richard F. Paige, *Agile Specification-Driven Development*, in J. Eckstein and H. Baumeister (Eds.): XP 2004, LNCS 3092, pp. 104–112, 2004

[3] Jonathan Kohl and Brian Marick, *Agile Tests as Documentation*, in C. Zannier et al. (Eds.): XP/Agile Universe 2004, LNCS 3134, pp. 198–199, 2004

[4] Pekka Klärck, Juha Rantanen, and Janne Härkönen, *Executable Requirements in Practice*, in P. Abrahamsson, M. Marchesi, and F. Maurer (Eds.): XP 2009, LNBIP 31, pp. 226–227, 2009

[5] John Barnes, *Programming in Ada 2005*. Addison-Wesley, 2006.

[6] Michael C. Feathers, *Working Effectively with Legacy Code*, Prentice Hall PTR, 2004

[8] http://www.open-do.org/projects/xreq/
[9] http://sogilis.com/go/xreq

Polymorphic Callbacks for Ada/C++ Bindings

Maciej Sobczak
CERN, CH-1211 Geneve 23, Switzerland; email: Maciej.Sobczak@cern.ch

Abstract

This article presents the programming technique that can be used to implement object-oriented callbacks in Ada bindings to C++ libraries that provide some form of notification mechanism. The content of this article is related to the industrial presentation that was given at the 15th International Conference on Reliable Software Technologies.

Keywords: language binding, object-oriented, Ada, C++.

1 The application context

The technique presented here was developed in the context of the YAMI4 library [1], which is a multi-language messaging solution for distributed systems. The multi-language aspect of this library means that several programming languages are supported by it, and also that some parts of the library are reused in the layered architecture, where a common (so-called "core") functionality is implemented in C++ and used as a foundation for the implementation of high-level communication services in Ada.

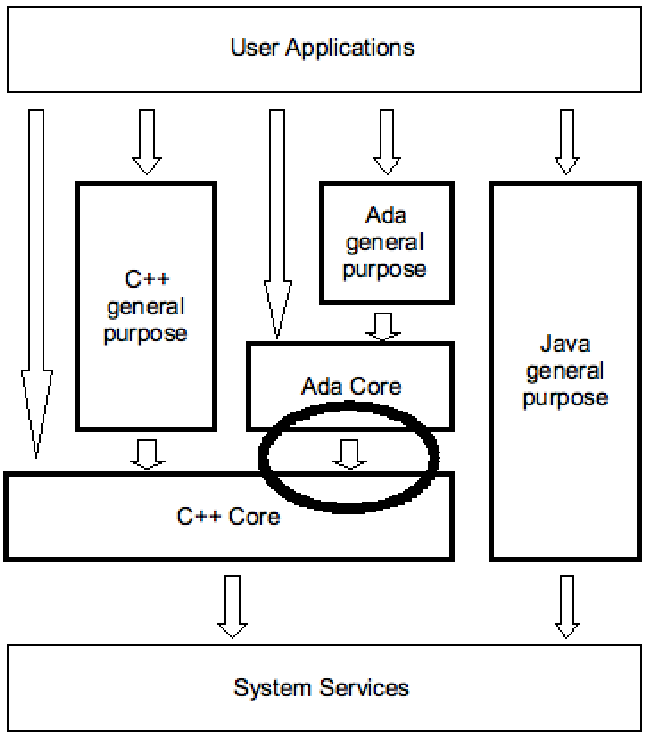

Figure 1. Architecture of YAMI4

The general architecture of the YAMI4 library is presented on Figure 1.

The YAMI4 messaging library uses callbacks to user-provided handlers in order to inform about important events - message progress notifications, incoming messages or connection status changes are possible events that are reported this way. For design reasons it is beneficial to expose such callback mechanisms in terms of interfaces that are implemented by the user, which gives them their object-oriented flavour - the *polymorphic callback* in this context means that the callback invocation itself uses dynamic dispatch in order to locate and execute the appropriate event handler implementation.

2 This is a general problem

It is important to note that the problem of callbacks is not particular to messaging systems. Similar interfacing patterns can be found in other systems as well - whether this is a web server passing requests to request handler, a database server calling its stored procedures, a GUI library invoking user actions or an alarm system passing event notifications to reactive components, the polymorphic callback can be a viable solution pattern. The problem has a relatively general nature and that's why it is possible to extract it in the form that is no longer related to the original application context.

The code examples presented here are self-contained and can be applied or extended in other application contexts.

To explain the problem in concrete terms, the following code presents the base C++ component that contains a simple notification mechanism with the callback interface:

```
//
// base.h
//
class Callback
{
public:
   virtual void call() = 0;
};

void registerCallback(Callback * c);
void fireAll();

//
// base.cpp
//
#include "base.h"
#include <cstdio>
#include <vector>

std::vector<Callback *> allCallbacks;

void registerCallback(Callback * c)
{
   std::puts("base: register callback");
```

```
      allCallbacks.push_back(c);
   }

   void fireAll()
   {
      std::puts("base: fire all!");
      std::vector<Callback *>::iterator it;
      for (it = allCallbacks.begin();
         it != allCallbacks.end(); ++it)
      {
         (*it)->call();
      }
   }
```

The problem here is to implement the Ada binding for this base component while preserving the object-oriented nature of the notification mechanism.

3 Ada binding with polymorphic callback

In order to implement the inter-language binding with callback interface it is worth to review the anatomy of the callback itself.

Object-oriented invocations, even though considered to be atomic or indivisible concepts, can be presented as pairs that combine an object and an action - the object represents the *who*, whereas the action represents the *what* of the invocation.

Incidentally both of these constituents can be expressed in terms of *pointers* or *addresses* - this makes them sufficiently low-level so that they can exist within the framework of the Ada/C++ binding, which in the Ada language standard is defined at the level of C language constructs. That is, since the C language is a *common denominator* of Ada and C++ for the purpose of language binding, expressing the callback as a pair of pointers is a necessary translation step - and that translation has to be performed on both Ada and C++ sides. Taking these additional translation layers into account, the final architecture for this solution consists of four layers:

- The Ada component that implements object-oriented callback handlers.
- The Ada translation layer that decomposes the Ada object-oriented callbacks into low-level *who* and *what* components and passes them down to the C++ layer.
- The C++ translation layer that exports its functionality in terms of C interface.
- The base C++ component that contains the notification mechanism.

Coming from the bottom up, the translation layer at the C++ side can be implemented as follows:

```
//
// wrapper.cpp
//
#include "base.h"

extern "C" typedef
void (*CallbackFunctionType)(void *);

class WrappedCallback : public Callback
{
public:
   WrappedCallback(
      CallbackFunctionType function, void * object)
   : f_(function), obj_(object) {}

   virtual void call()
   {
      // call into the Ada translator procedure
      f_(obj_);
   }

private:
   CallbackFunctionType f_;
   void * obj_;
};

extern "C" void wrapped_registerCallback(
   void * function_addr, void * object)
{
   // brute-force conversion from raw procedure
   // address obtained from Ada
   // to function pointer that is useable
   // at the C++ level
   union
   {
      void * raw_pointer;
      CallbackFunctionType function_pointer;
   } converter;

   converter.raw_pointer = function_addr;
   CallbackFunctionType function =
      converter.function_pointer;

   registerCallback(
      new WrappedCallback(function, object));
}

extern "C" void wrapped_fireAll()
{
   fireAll();
}
```

The wrapper layer above consists of three definitions with the "C" convention - the callback function pointer type, which can carry the translated address of the action to be executed on the Ada side, and two wrappers for the notification mechanism.

The most important part here is the registration function, which has to convert the procedure address passed by the Ada layer to the genuine function pointer. Interestingly, converting raw pointers to function pointers is not obvious in C++ and above the union is used as the most low-level way to achieve that goal. Together with the class wrapper that implements the base interface, this translation layer can pass callback invocations up to the higher layers, where they are handled by Ada code.

The translation layer in Ada has the form of the following package:

```
--
-- callbacks.ads
--
```

```
package Callbacks is

   type Callback is interface;
   type Callback_Access is access all Callback'Class;

   procedure Call (Self : in Callback) is abstract;

   procedure Register_Callback
     (C : in Callback_Access);
   procedure Fire_All;

end Callbacks;

--
--  callbacks.adb
--
with System.Address_To_Access_Conversions;

package body Callbacks is

   subtype Void_Ptr is System.Address;

   package Conversions is
     new System.Address_To_Access_Conversions
    (Object => Callback'Class);

   --  helper translator,
   --  will be directly called by the C++ wrapper:
   procedure Callback_Translator (Obj : in Void_Ptr);
   pragma Convention (C, Callback_Translator);

   procedure Callback_Translator (Obj : in Void_Ptr) is
     Callback_Handler : Callback_Access :=
       Callback_Access (Conversions.To_Pointer (Obj));
   begin
     --  actual dispatching call to the Ada implementation:
     Callback_Handler.all.Call;
   end Callback_Translator;

   procedure Register_Callback
    (C : in Callback_Access) is

     procedure Wrapped_Register_Callback
       (Fun : in Void_Ptr; Obj : in Void_Ptr);
     pragma Import (C, Wrapped_Register_Callback,
               "wrapped_registerCallback");
   begin
     Wrapped_Register_Callback
       (Callback_Translator'Address,
        Conversions.To_Address
         (Conversions.Object_Pointer (C)));
   end Register_Callback;

   procedure Fire_All is
     procedure Wrapped_Fire_All;
     pragma Import (C, Wrapped_Fire_All,
                         "wrapped_fireAll");
   begin
     Wrapped_Fire_All;
   end Fire_All;

end Callbacks;
```

As can be seen above, the package specification is already "digestible" as a high-level component that can be directly used in the same way as the base component in C++. The actual translation work is being done in the package body, where the object-oriented callbacks are decomposed into two access values - the *who*, which directly corresponds to the handler object, and the *what*, or *action* component, which is a trampoline subprogram that takes the object address from the C++ layer and uses it for a final dispatching call to the Ada callback handler.

The example Ada program that uses the whole mechanism can look like this:

```
--
--  example.adb
--
with Ada.Text_IO; use Ada.Text_IO;
with Callbacks;

procedure Example is

   type Some_Callback is
     new Callbacks.Callback with null record;
   overriding procedure Call (Self : in Some_Callback);

   overriding procedure Call
     (Self : in Some_Callback) is
   begin
     Put_Line ("Ada: Some Callback called");
   end Call;

   type Other_Callback is
     new Callbacks.Callback with null record;
   overriding procedure Call (Self : in Other_Callback);

   overriding procedure Call
     (Self : in Other_Callback) is
   begin
     Put_Line ("Ada: Other Callback called");
   end Call;

   SC : aliased Some_Callback;
   OC : aliased Other_Callback;

begin
   Put_Line ("Ada: registering callbacks");
   Callbacks.Register_Callback
     (SC'Unchecked_Access);
   Callbacks.Register_Callback
     (OC'Unchecked_Access);

   Put_Line ("Ada: fire all!");
   Callbacks.Fire_All;
end Example;
```

For completeness, the following compiler invocations can be used to build the final executable on Linux or Mac:

```
$ g++ -c base.cpp
$ g++ -c wrapper.cpp
$ gnatmake example -largs wrapper.o base.o -lstdc++
```

4 Maintenance impact

An important property of this solution is that it is not intrusive with relation to the base component, which here was written in C++. This reflects a typical scenario where a C++ library is wrapped by a binding layer in Ada. The technique presented here allows to implement an efficient binding without introducing any reverse dependency between C++ and Ada and without modifying the original base component in any way, at the same time preserving the object-oriented nature of the notification mechanism.

Another advantage of this technique is that each layer can be replaced with different implementation - in particular, the low-level base component can be rewritten in Ada with no impact on the user code, as the Ada specification already has the appropriate high-level structure. Other combinations of languages and their bindings are also possible with little on no impact on the existing components.

5 Inter-language issues

It can be interesting to note that in this implementation of object-oriented (polymorphic) callbacks the two low-level callback constituents - that is, the two pointers that are passed between layers written in two languages - have different usage paths.

The object pointer, which comes from the access value to the Ada callback handler instance, is created at the Ada level, passed down to C++ for storage only and then - when the notification is performed - passed up to Ada again where the pointer is converted back to access value. In other words, the access value is obtained and used within the context of the same programming language and the other language is given that value only for temporary storage.

Contrary to this, the action pointer, which for the binding purpose is obtained from the trampoline procedure in the Ada translation layer, has a somewhat different usage path: it is obtained at the Ada level, passed down to C++, reconstructed as a function pointer and used right there to invoke the trampoline procedure in Ada. That is, the access value is obtained in the context of one programming language but actually used in the context of another. It might be interesting to note that the Ada language standard [2] precisely defines the mechanics of passing subprogram access values to external components, whereas the meaning of pointer value that is received at the C++ side [3] is a bit less clear. The conversion that is employed for this value at the lower level can be considered to be a weak point of this solution and a potential portability issue - but the truth is that in order to effectively bind two programming languages it is not possible to rely on language standards in isolation and a pair of "friendly" compilers is needed anyway; such a pair can make up for missing standard provisions and that is particularly true for GNAT and g++, which happen to be part of the same compiler toolchain.

6 Performance considerations

The multi-layer architecture of this solution can raise performance-related questions - what is the cost of such dual translation?

Even though no strict measurements have been done to answer such questions in the quantitative way, it is safe to assume that the solution presented here is very efficient. The reason for this is that both translation layers perform only type conversions and are very cheap in terms of object code. In relation to pure-C++ implementation the Ada/C++ binding solution adds an overhead of a single subprogram call and a single dispatching operation call.

7 Conclusions

The technique presented in this article allows to efficiently implement language bindings for C++ libraries that provide notification functionality in the form of object-oriented (that is, involving run-time dispatching invocations) callbacks. This technique has been successfully used in a real project in the domain of communication and messaging.

The examples presented here are of general nature and can be deployed in other application domains. Obviously, some code extensions will be necessary for callbacks involving parameters or to properly handle error propagation – these details were not within the focus of this article and interested readers are invited to consult the source code of the YAMI4 library [1], where such issues have been resolved fully and in a realistic context. The author is also open to any related questions.

References

[1] YAMI4 homepage: http://www.inspirel.com/yami4

[2] Ada Reference Manual, ISO/IEC 8652:2007(E) Ed. 3, B.1 Interfacing Pragmas

[3] C++ Standard (ISO/IEC 14882:1998(E)), 4.10 Pointer conversions

Ada Gems

The following contributions are taken from the AdaCore Gem of the Week series. The full collection of gems, discussion and related files, can be found at http://www.adacore.com/category/developers-center/gems/.

Gems #71-#76: Tokeneer Discovery Lessons

Dean Kuo and Angela Wallenburg, Altran Praxis

Dates: 5 October 2009-14 December 2009

Abstract: In previous Gems [1], we saw how to create a SPARK project in GPS, and how to run the Examiner and Simplifier tools to verify various properties of a simple linear search function. In this series of Gems on SPARK, we will explore SPARK capabilities in greater depth, using source code from Tokeneer, an NSA-funded, highly secure biometric software system.

Note: This series of Gems makes references to SPARK and the Tokeneer project. More about both can be found visiting *http://www.adacore.com/home/products/sparkpro/tokeneer/*.

Further, an introduction to SPARK and contracts in SPARK is available at *http://www.adacore.com/home/products/sparkpro/tokeneer/discovery/* and at *http://www.adacore.com/home/products/sparkpro/tokeneer/discovery/lesson_contracts/*.

Lesson 1 – Improper initialization

As stated in the SANS Common Weakness Enumeration (http://cwe.mitre.org/top25/#CWE-665): "If you don't properly initialize your data and variables, an attacker might be able to do the initialization for you, or extract sensitive information that remains from previous sessions. When those variables are used in security-critical operations, such as making an authentication decision, then they could be modified to bypass your security. Incorrect initialization can occur anywhere, but it is probably most prevalent in rarely-encountered conditions that cause your code to inadvertently skip initialization, such as obscure errors." Improper initialization is listed as one of the top twenty-five most dangerous programming errors by the SANS Institute. How would you find those? How much would your compiler help?

The SPARK Toolset identifies all variables that have not been initialized prior to being read. And if the SPARK automatic source code analysis finds no uninitialized variables, then there really are none!

In this Gem, we will first inject a couple of initialization errors into some of the Tokeneer code and then show how you can use the SPARK tools to find those errors.

Step-by-Step Instructions

Let us start by first running the Examiner, one of the SPARK tools, on some of the Tokeneer code in the correct version. Then we will inject an uninitialized variable error in the same code and see how the Examiner's data-flow analysis finds it for us.

Step 1: Run the Examiner on the correct version of the code

The code below is from the procedure GetStartAndEndTimeFromFile in auditlog.adb.

```
264  procedure GetStartAndEndTimeFromFile
265    (TheFile     : in out File.T;
266     Description :    out AuditTypes.DescriptionT)
267    --# global in out AuditSystemFault;
268    --# derives AuditSystemFault,
269    --#         TheFile         from *,
270    --#                              TheFile &
271    --#         Description     from TheFile;
272  is
273     OK : Boolean;
274     FirstTime : Clock.TimeTextT;
275     LastTime  : Clock.TimeTextT;
276     TimeCount : Natural; -- type Natural to match
                                  formal parameter
277                      -- in call to GetString
278     TimeOK    : Boolean := True;
279
   ...
311  ------------------------------------------------------------------
312  -- begin GetStartAndEndTimeFromFile
313  ------------------------------------------------------------------
314  begin
315
316     FirstTime := Clock.PrintTime(Clock.ZeroTime);
317     LastTime := Clock.PrintTime(Clock.ZeroTime);
   ...
358     Description := ConvertTimesToText;
359
360  end GetStartAndEndTimeFromFile;
```

Run the Examiner on auditlog.adb and notice that it reports no errors.

Step 2: Inject an initialization error and find it using the Examiner

Let's introduce two types of uninitialized variables into the code – variables that are never initialized and variables that may not be initialized.

Lines 316 and 317 initialize the variables FirstTime and LastTime. Put these two lines of code inside an if statement so that the variables are only initialized if the variable OK is true. The modified code is shown below.

```
316  if OK then
317     FirstTime := Clock.PrintTime(Clock.ZeroTime);
318     LastTime := Clock.PrintTime(Clock.ZeroTime);
319  end if;
```

[1] Ada Gems #68 and #69, also published in the Ada User Journal Vol. 30, n. 3, September 2009

The changes introduce a number of improper initialization errors:

- The variable OK is not initialized before it is read;
- The variables FirstTime and LastTime may not be initialized;
- The uninitialized variables FirstTime, LastTime and OK may then influence the expression that assigns a value to the variable Description -- the variable OK is indirectly used in the expression.

Rerun the Examiner and notice that the Examiner identifies all uninitialized variables as errors, and it differentiates between variables that are not initialized and variables that may not be initialized:

```
auditlog.adb:316:10:
  Flow Error  20 - Expression contains reference(s) to
        variable OK which has an undefined value.
auditlog.adb:316:10:
  Flow Error  22 - Value of expression is invariant.
auditlog.adb:360:7:
  Flow Error 504 - Statement contains reference(s) to
        variable FirstTime, which may have an
        undefined value.
auditlog.adb:360:7:
  Flow Error 504 - Statement contains reference(s) to
        variable LastTime, which may have an
        undefined value.
auditlog.adb:362:8:
  Flow Error 602 - The undefined initial value of OK may be
        used in the derivation of Description.
auditlog.adb:362:8:
  Flow Error 602 - The undefined initial value of FirstTime
        may be used in the derivation of Description.
auditlog.adb:362:8:
  Flow Error 602 - The undefined initial value of LastTime
        may be used in the derivation of Description.
```

Notice also that the compiler already warns against the simplest of these improper initialization errors:

```
auditlog.adb:316:04: warning: "OK" may be referenced
        before it has a value
```

Step 3: Find an array index initialization error

Another common error is accessing an array element with an uninitialized variable as the index. In this example we will inject such an error and demonstrate that the Examiner also finds these errors.

The code below is from the procedure DeleteLogFile in auditlog.adb.

```
512  procedure DeleteLogFile ( Index : LogFileIndexT)
513  --# global in out AuditSystemFault;
514  --#        in out LogFiles;
515  --#        in out LogFilesStatus;
516  --#        in out LogFileEntries;
517  --# derives AuditSystemFault,
518  --#         LogFiles        from *,
519  --#                              LogFiles,
520  --#                              Index &
521  --#         LogFilesStatus,
522  --#         LogFileEntries   from *,
523  --#                              Index;
524  is
525    OK : Boolean;
526    TheFile : File.T;
527  begin
528
529    TheFile := LogFiles (Index);
     ...
543  end DeleteLogFile;
```

Line 529 of the code accesses the Indexth element of the array LogFiles.

Declare a new variable I of type LogFileIndexT and replace Index, on line 529, with I. The modified code is shown below.

```
512  procedure DeleteLogFile (Index: LogFileIndexT)
     ...
524  is
525     OK : Boolean;
526     TheFile : File.T;
527     I : LogFileIndexT;
528  begin
529
530     TheFile := LogFiles (I);
```

Run the Examiner on the file auditlog.adb and notice that it reports that the variable I has not been initialized:

```
auditlog.adb:530:18:
  Flow Error  20 - Expression contains reference(s) to
        variable I which has an undefined value.
auditlog.adb:544:8:
  Flow Error  32 - The variable I is neither imported nor
        defined.
auditlog.adb:544:8:
  Flow Error  50 - AuditSystemFault is not derived from the
        imported value(s) of Index.
auditlog.adb:544:8:
  Flow Error 602 - The undefined initial value of I may be
        used in the derivation of AuditSystemFault.
auditlog.adb:544:8:
  Flow Error 602 - The undefined initial value of I may be
        used in the derivation of LogFiles.
```

Again, in this very simple case, the compiler is able to spot the error too:

```
auditlog.adb:530:07: warning: variable "I" is read but never
        assigned
```

Although warnings from the compiler and warnings from the Examiner can sometimes overlap, one should keep in mind that the compiler warnings are heuristic, while the Examiner will warn against all possible improper initialization errors.

Summary

We have seen that the the SPARK tools can verify that a SPARK program is free from improper input validation errors. In the next Gem, we will study how SPARK handles a related class of errors — identifying ineffective statements.

Lesson 2 – How to identify ineffective statements.

Every statement should have a purpose. An ineffective statement has no effect on any output variable and therefore has no effect on the behaviour of the code. The presence of ineffective statements reduces the quality and the maintainabiliy of the code. The SPARK Toolset identifies all ineffective statements.

In this Gem, we show how the SPARK Toolset finds ineffective statements to ensure that SPARK programs are free from them. We will inject an ineffective statement into the Tokeneer code and use the Examiner to locate it.

Step 1: Inject an ineffective statement

We will inject an ineffective statement into the implementation of the function NextListIndex in auditlog.adb.

```
189  function NextListIndex(Value : LogFileIndexT)
              return  LogFileIndexT
190  is
191     Result : LogFileIndexT;
192  begin
193     if Value = LogFileIndexT'Last then
194        Result := LogFileIndexT'First;
195     else
196        Result := Value + 1;
197     end if;
198     return Result;
199  end NextListIndex;
```

Let's modify the above code by adding the new variable Result_Tmp of type LogFileTypeT (line 191) and change line 196 so that Value + 1 is assigned to the variable Result_Tmp instead of Result (see code below).

```
189  function NextListIndex(Value : LogFileIndexT)
              return  LogFileIndexT
190  is
191     Result, Result_Tmp: LogFileIndexT;
192  begin
193     if Value = LogFileIndexT'Last then
194        Result := LogFileIndexT'First;
195     else
196        Result_Tmp := Value + 1;
197     end if;
198     return Result;
199  end NextListIndex;
```

The statement on line 196 is ineffective because Result_Tmp is never used. Furthermore, the value for Result may be undefined when Value /= LogFileIndexT'Last.

Step 2: See how the Examiner finds the problem

Run the Examiner for auditlog.adb and, as expected, it finds the ineffective statement as well as the possible undefined value for Result.

```
auditlog.adb:196:10:
  Flow Error  10 - Ineffective statement.
auditlog.adb:198:14:
  Flow Error 501 - Expression contains reference(s) to
          variable Result, which may have an undefined value.
auditlog.adb:199:8:
  Flow Error  33 - The variable Result_Tmp is neither
          referenced nor exported.
auditlog.adb:199:8:
  Flow Error 602 - The undefined initial value of Result may
          be used in the derivation of the function value.
```

Summary

In this Gem we have seen an example of the SPARK tools finding an ineffective statement. The SPARK tools will find all ineffective statements. In the next Gem we will study Input Validation.

Lesson 3 – How to validate input.

Input validation ensures that your program's input conforms to expectations – for example, to ensure that the input has the right type. But validation requirements can be much more complicated than that. Incorrect input validation can lead to security and safety problems since many applications live in a "hostile" environment and the input might be constructed by an attacker. "It's the number one killer of healthy software…" according to the CWE/SANS list of the top twenty-five most dangerous programming errors.

For example, consider the following few lines of code from the original release of the Tokeneer code:

```
233    if Success and then
234       (RawDuration * 10 <= Integer(DurationT'Last)
          and
235        RawDuration * 10 >= Integer(DurationT'First))
          then
236       Value := DurationT(RawDuration * 10);
237    else
```

This code has a check that the input RawDuration is in the right range before the value is updated – an example of so called defensive coding, according to the advice from the software experts who compiled the list of dangerous programming errors. Can you see the problem with this code?

The SPARK tools will identify a serious defect in this code, which could impact on security.

In this Gem, the above code will be investigated using the SPARK tools. This Gem shows the challenges in ensuring the absence of input validation errors, and the benefits of using SPARK to do so.

Step-by-Step Instructions

First, we will familiarize ourselves with two more advanced SPARK tools by running them on the correct version of the code. Then we inject the defect shown above into the Tokeneer code, rerun the SPARK tools, and interpret the results.

Step 1: Analyse the Correct Version of the Code

The correct lines of code are the following:

```
233    if Success and then
234       (RawDuration <= Integer(DurationT'Last) / 10
          and
235        RawDuration >= Integer(DurationT'First) / 10)
          then
236       Value := DurationT(RawDuration * 10);
237    else
```

Analyse Tokeneer using the following steps:

- Run the Examiner on the file configdata.adb.
- Run the Simplifier, a more advanced tool in the SPARK tool suite, which tries to show the absence of certain run-time errors by theorem proving.
- Run POGS, which gives a summary of the verification just performed.

```
2750   VCs for procedure_readduration :
2751   -----------------------------------------------------------------------------
2752        |       |                        |  -----Proved In-----  |       |       |
2753    #   | From  | To                     | vcg | siv | plg | prv | False | TO DO |
2754   -----------------------------------------------------------------------------
2755    1   | start | rtc check @ 220        |     | YES |     |     |       |       |
2756    2   | start | rtc check @ 221        |     | YES |     |     |       |       |
2757    3   | start | rtc check @ 221        |     | YES |     |     |       |       |
2758    4   | start | rtc check @ 233        |     | YES |     |     |       |       |
2759    5   | start | rtc check @ 237        |     | YES |     |     |       |       |
2760    6   | start | rtc check @ 243        |     | YES |     |     |       |       |
2761    7   | start | rtc check @ 243        |     | YES |     |     |       |       |
2762    8   | start |    assert @ finish     | YES |     |     |     |       |       |
2763    9   | start |    assert @ finish     | YES |     |     |     |       |       |
2764    10  | start |    assert @ finish     | YES |     |     |     |       |       |
2765    11  | start |    assert @ finish     | YES |     |     |     |       |       |
2766   -----------------------------------------------------------------------------
```

Figure 1 Results from the analysis of the correct procedure ReadDuration

Now let us inspect the verification summary, where an overall summary of the Tokeneer verification is given. It shows that there are no errors which is expected as we ran the tools on the correct version of the code. Furthermore, it shows a number of tables describing details for the verification. For example, lines 2750 to 2766 of core.sum (Figure 1) are the results from the analysis of the procedure ReadDuration.

Note that the columns False and TO DO are empty, which means that the SPARK tools found no errors in the verification of procedure ReadDuration.

Step 2: Inject Erroneous Input Validation Code

Now, replace lines 234 and 235 in the file configdata.adb with the erroneous code:

```
233   if Success and then
234     (RawDuration * 10 <= Integer(DurationT'Last)
        and
235      RawDuration * 10 >= Integer(DurationT'First))
        then
236     Value := DurationT(RawDuration * 10);
237   else
```

Essentially this code concerns the validation of an input – an integer value RawDuration – that is read from a file, and is expected to be in the range 0..200 seconds before it is converted into a number of tenths of seconds in the range 0..2000.

Step 3: Re-Analyse the Faulty Code

Re-analyse Tokeneer. The results from the analysis of the procedure ReadDuration is shown in Figure 2.

Notice that this time there is one YES in the TO DO column. The SPARK Toolset has detected a potential problem with the procedure ReadDuration.

Step 4: Investigate the Verification Output

Now, let us have look into what the error that the SPARK tools have found really means.

The file readduration.siv contains the Simplifier's analysis of the procedure. Lines 38 to 55 (rigth column) of the file show the potential problem the SPARK Toolset has identified. The Simplifier, on line 54, is trying to check no arithmetic overflow errors will occur when evaluating the expression RawDuration * 10 – that is, when Success is True then RawDuration * 10 >= -2147483648 (Integer'First) and RawDuration * 10 <= 2147483648 (Integer'Last).

```
38   procedure_readduration_4.
39   H1:   rawduration__1 >= - 2147483648 .
40   H2:   rawduration__1 <= 2147483647 .
  ...
52          ->
53   C1:   success__1 -> rawduration__1 * 10 >=
            - 2147483648 and rawduration__1 *
54          10 <= 2147483647 .
55
```

Here the SPARK theorem prover is trying to prove that RawDuration times 10 is within the limits of Integer, assuming only that it was within the limits before it was multiplied by 10. This should not be possible to prove. Think about a scenario where Tokeneer was given an input floppy

```
2750   VCs for procedure_readduration :
2751   -----------------------------------------------------------------------------
2752        |       |                        |  -----Proved In-----  |       |       |
2753    #   | From  | To                     | vcg | siv | plg | prv | False | TO DO |
2754   -----------------------------------------------------------------------------
2755    1   | start | rtc check @ 220        |     | YES |     |     |       |       |
2756    2   | start | rtc check @ 221        |     | YES |     |     |       |       |
2757    3   | start | rtc check @ 221        |     | YES |     |     |       |       |
2758    4   | start | rtc check @ 233        |     |     |     |     |       |  YES  |
2759    5   | start | rtc check @ 237        |     | YES |     |     |       |       |
2760    6   | start | rtc check @ 243        |     | YES |     |     |       |       |
2761    7   | start | rtc check @ 243        |     | YES |     |     |       |       |
2762    8   | start |    assert @ finish     | YES |     |     |     |       |       |
2763    9   | start |    assert @ finish     | YES |     |     |     |       |       |
2764    10  | start |    assert @ finish     | YES |     |     |     |       |       |
2765    11  | start |    assert @ finish     | YES |     |     |     |       |       |
2766   -----------------------------------------------------------------------------
```

Figure 2 Results from the analysis of the faulty procedure ReadDuration

```
VCs for procedure_addelementtocurrentfile :
-------------------------------------------------------------------------
      |       |                     |  -----Proved In-----  |       |       |
 #    | From  | To                  | vcg | siv | plg | prv | False | TO DO |
-------------------------------------------------------------------------
 1    | start | rtc check @ 781     |     | YES |     |     |       |       |
 2    | start | rtc check @ 782     |     | YES |     |     |       |       |
 3    | start | rtc check @ 788     |     | YES |     |     |       |       |
 4    | start | rtc check @ 790     |     | YES |     |     |       |       |
 5    | start |    assert @ finish  |     | YES |     |     |       |       |
-------------------------------------------------------------------------
```

Figure 3 Results from the analysis of the correct procedure AddElementToCurrentFile

disk where RawDuration was set to 1000000000. Both the assumptions H1 and H2 would be true, but C1 – the conclusion – would be false!

This is a serious defect since a malicious user holding the "security officer" role can deliberately attack the system by supplying a file that contains a malformed configuration data file – one that contains a value for RawDuration that is greater than Integer'Last/10.

Summary

In SPARK, developers need to be explicit about the intended input and output of program components. This has the benefit of the SPARK tools being able to automatically find defects that are hard to prevent, hard to detect, and with important security consequences.

Lesson 4 – verify application-specific safety and security properties.

In this Gem, we will see how the SPARK tools detect any differences between a program's intended behaviour, as specified in its contract, and its actual behaviour, as implemented in the code. Thus the SPARK tools can be used either to find defects in the contract, to find defects in the implementation, to find defects in both, or to show conformance between intended and actual behaviour.

Step-by-Step Instructions

Step 1: Analyse the Correct Version of the Code

The precondition for procedure AddElementToCurrentFile in auditlog.adb (lines 772 – 774 in the code below) specifies that the array element LogFileEntries (CurrentLogFile) is less than MaxLogFileEntries and the postcondition specifies that the array element is incremented by 1.

```
751   procedure AddElementToCurrentFile
752     (ElementID   : in    AuditTypes.ElementT;
753      Severity    : in    AuditTypes.SeverityT;
754      User        : in    AuditTypes.UserTextT;
755      Description : in    AuditTypes.DescriptionT)
756   --# global in     Clock.Now;
757   --#        in     CurrentLogFile;
758   --#        in out AuditSystemFault;
759   --#        in out LogFiles;
760   --#        in out LogFileEntries;
761   --# derives AuditSystemFault,
762   --#         LogFiles       from *,
763   --#                             Description,
764   --#                             LogFiles,
765   --#                             Clock.Now,
766   --#                             ElementID,
767   --#                             Severity,
768   --#                             User,
769   --#                             CurrentLogFile &
770   --#         LogFileEntries  from *,
771   --#                             CurrentLogFile;
772   --# pre LogFileEntries(CurrentLogFile) <
                                   MaxLogFileEntries;
773   --# post LogFileEntries(CurrentLogFile) =
774   --#        LogFileEntries~(CurrentLogFile) + 1;
775
776   is
777     TheFile : File.T ;
778   begin
779     TheFile := LogFiles (CurrentLogFile);
780     AddElementToFile
781       (TheFile => TheFile,
782        ElementID   => ElementID,
783        Severity    => Severity,
784        User        => User,
785        Description  => Description);
786     LogFiles (CurrentLogFile) := TheFile;
787
788     LogFileEntries(CurrentLogFile) :=
                LogFileEntries(CurrentLogFile) + 1;
789   end AddElementToCurrentFile;
```

Analyse Tokeneer with the SPARK Toolset. The result of the analysis shows (Figure 3) that the SPARK Toolset has identified no problems with the code in the procedure AddElementToCurrentFile – the columns False and TO DO are empty.

Step 2: Change the Contract – But not the Implementation

Let us change the postcondition so the procedure's contract no longer matches its implementation. The SPARK Toolset will then detect the inconsistency. Change the postcondition to specify that the array element LogFileEntries(CurrentLogFile) should be incremented by 10. The modified code is shown below.

```
772   --# pre LogFileEntries(CurrentLogFile) <
                             MaxLogFileEntries;
773   --# post LogFileEntries(CurrentLogFile) =
774   --#      LogFileEntries~(CurrentLogFile) + 10;
775
```

Step 3: Re-Analyse and Study the Results

Re-analyse Tokeneer. The results of the analysis for the procedure AddElementToCurrentFile has changed – the columns False and TO DO are no longer empty (see Figure 4).

The SPARK Toolset has identified a potential problem with the code. The problem is that the procedure's contract and implementation don't match.

Step 4: Revert the Contract – But Change the Implementation

Undo the changes to the postcondition and change line 788 so that the input value is preserved.

```
659  VCs for procedure_addelementtocurrentfile :
660  -------------------------------------------------------------------------
661       |       |                     |  -----Proved In-----  |       |       |
662  #    | From  | To                  | vcg | siv | plg | prv | False | TO DO |
663  -------------------------------------------------------------------------
664   1   | start | rtc check @ 781     |     | YES |     |     |       |       |
665   2   | start | rtc check @ 782     |     | YES |     |     |       |       |
666   3   | start | rtc check @ 788     |     | YES |     |     |       |       |
667   4   | start | rtc check @ 790     |     | YES |     |     |       |       |
668   5   | start |    assert @ finish  |     |     |     |     |  YES  |       |
669  -------------------------------------------------------------------------
```

Figure 4 Results from the analysis of the inconsitency in contract of AddElementToCurrentFile

```
774       --# pre LogFileEntries(CurrentLogFile) <
                                MaxLogFileEntries;
775       --# post LogFileEntries(CurrentLogFile) =
776       --#        LogFileEntries~(CurrentLogFile) + 1
          ...
791       LogFileEntries(CurrentLogFile) :=
                 LogFileEntries(CurrentLogFile) + 0;
792     end AddElementToCurrentFile;
```

Re-analyse Tokeneer. The analysis of the procedure is unchanged, as the implementation, like previously, does not match the procedure's contract.

Step 5: Strengthen the Contract

Revert the code to its original state.

In SPARK, we can strengthen the procedure's contract and say more about the properties of the procedure. Let's add extra code assigning the value 10 to the first element of the array LogFileEntries if CurrentLogFile is not LogFileIndexType' First (lines 789 – 791 below).

```
788      LogFileEntries(CurrentLogFile) :=
                LogFileEntries(CurrentLogFile) + 1;
789      if CurrentLogFile /= LogFileIndexType'First then
790         LogFileEntries(LogFileIndexType'First) := 10;
791      end if;
792    end AddElementToCurrentFile;
```

Re-analyse Tokeneer and notice that no errors are reported, as the procedure's implementation is not inconsistent with its contract. The postcondition says nothing about the effects of the procedure on any of the array elements except the one indexed by CurrentLogEntry.

We can strengthen the postcondition (lines 775 and 776 below) to specify that only the entry indexed by CurrentLogFile is incremented and all other elements remain unchanged.

```
772  --# pre LogFileEntries(CurrentLogFile) <
            MaxLogFileEntries;
773  --# post LogFileEntries =
            LogFileEntries~[CurrentLogFile =>
            LogFileEntries~(CurrentLogFile)+1];
```

Re-analyse Tokeneer and notice, on lines 659 to 671, that the mismatch between the implementation and contract has been detected (Figure 5).

Summary

This Gem demonstrates that the more precise the specification, the more bugs the SPARK Toolset can detect. The use of the SPARK Toolset during development to verify code is, in our experience, more effective than compiling and testing, since the analysis is for all input data and not just a few specific test cases.

Lesson 5 – Deal with overflow errors.

An overflow error occurs when the capacity of a device is exceeded. Overflow errors are a source of quality and security concerns. For instance, when an arithmetic overflow occurs, a calculated value does not fit in its specified size, and the calculation (and the program) just stops. Buffer overflow happens when a process stores data in a buffer outside of the memory that the programmer set aside for it. Buffer overflow errors are widely known to present a vulnerability to malicious hackers, who might exploit the error to sneak their own code onto a victim's disk, storing it outside of the intended buffer.

The SPARK tools detect all potential arithmetic and buffer overflow errors. In the Gem about input validation, we saw an example of an arithmetic overflow. In this Gem, we will study how the SPARK tools find a buffer overflow error that we have injected into the Tokeneer code.

Step-by-Step Instructions

We will introduce a buffer overflow error into auditlog.adb and show how the SPARK Toolset detects it.

Step 1: Inject a Buffer Overflow Error

The procedure AddElementToCurrentFile increments the element indexed by CurrentLogFile in the array

```
657  VCs for procedure_addelementtocurrentfile :
658  -------------------------------------------------------------------------
659       |       |                     |  -----Proved In-----  |       |       |
660   #   | From  | To                  | vcg | siv | plg | prv | False | TO DO |
661  -------------------------------------------------------------------------
662   1   | start | rtc check @ 783     |     | YES |     |     |       |       |
663   2   | start | rtc check @ 784     |     | YES |     |     |       |       |
664   3   | start | rtc check @ 790     |     | YES |     |     |       |       |
665   4   | start | rtc check @ 792     |     | YES |     |     |       |       |
666   5   | start | rtc check @ 794     |     | YES |     |     |       |       |
667   6   | start |    assert @ finish  |     |     |     |     |       |  YES  |
668   7   | start |    assert @ finish  |     | YES |     |     |       |       |
669  -------------------------------------------------------------------------
```

Figure 5 Results from the analysis of the mismatch in AddElementToCurrentFile

```
VCs for procedure_addelementtocurrentfile :
---------------------------------------------------------------------------
      |        |                   |  -----Proved In-----  |       |       |
#     | From   | To                | vcg | siv | plg | prv | False | TO DO |
---------------------------------------------------------------------------
 1    | start  | rtc check @ 781   |     | YES |     |     |       |       |
 2    | start  | rtc check @ 782   |     | YES |     |     |       |       |
 3    | start  | rtc check @ 788   |     | YES |     |     |       |       |
 4    | start  | rtc check @ 790   |     |     |     |     |       |  YES  |
 5    | start  |    assert @ finish|     | YES |     |     |       |       |
---------------------------------------------------------------------------
```

Figure 6 Results from the analysis of the mismatch in AddElementToCurrentFile

LogFileEntries – see below.

```
775 --# post LogFileEntries(CurrentLogFile) =
776 --#        LogFileEntries~(CurrentLogFile) + 1;
    ...
790   LogFileEntries(CurrentLogFile) :=
           LogFileEntries(CurrentLogFile) + 1;
```

Change the code on line 790 and the postcondition on 775 to 776 so the procedure copies the value in the CurrentLogFile+1th element into the CurrentLogFileth element of the array. The modified code is shown below.

```
774 --# pre LogFileEntries(CurrentLogFile) <
               MaxLogFileEntries;
775 --# post LogFileEntries(CurrentLogFile) =
776 --#        LogFileEntries~(CurrentLogFile+1);
    ...
790 LogFileEntries(CurrentLogFile) :=
          LogFileEntries(CurrentLogFile+1) ;
```

Step 2: Analyse and Study the Verification Output

Analyse Tokeneer. The SPARK Toolset identifies (see Figure 6), that there is a potential problem with the procedure AddElementToCurrentFile.

The Simplifier failed to show that CurrentLogFile+1 is always within range of the index type, because it is not true – it goes outside its range when CurrentLogFile = LogFileIndexType' Last, which then causes a buffer overflow.

Summary

Buffer overflow errors are common and present a security vulnerability. The SPARK Toolset can verify that a SPARK program is free from arithmetic as well as buffer overflow errors. In this Gem we have seen how the SPARK tools can be used to detect a buffer overflow error. In the next Gem, we will see how SPARK can be used in Ensuring Secure Information Flow.

Lesson 6 – Ensure secure information flow.

Error message information leak occurs when secure data is leaked, through error messages, to unauthorised users, and is one of the top twenty-five most dangerous programming errors according to SANS Institute. The general problem is ensuring that information flow adheres to certain policies — for example, certain data should never be written in an error message to a log file that may be accessible by unauthorised users.

The objective of this Gem is to demonstrate that the Examiner detects information flow violations.

Step-by-Step Instructions

Step 1: Study a Contract from an Information Flow Perspective

The code below is from the procedure Verify in bio.adb. The out variable MatchResult returns the result of whether a person's fingerprint matched their template. The local variable NumericReturn is set to the enumerated value BioApiOk if the fingerprint successfully matched; otherwise it returns an error code.

When a match is unsuccessful, a log record is written including the variable NumericReturn, which is derived from the person's Template.

```
221 procedure Verify(Template    : in
                          IandATypes.TemplateT;
222          MaxFAR      : in   IandATypes.FarT;
223          MatchResult  :  out
                          IandATypes.MatchResultT;
224          AchievedFAR  :  out
                          IandATypes.FarT)
    ...
230 --# derives AuditLog.State,
231 --#       AuditLog.FileState from AuditLog.State,
232 --#                        AuditLog.FileState,
233 --#                        Template,
234 --#                        Clock.Now,
235 --#                        ConfigData.State,
236 --#                        Interface.Input &
    ...
242 is
243   NumericReturn : BasicTypes.Unsigned32T;
244 begin
245   Interface.Verify(Template     => Template,
246              MaxFAR       => MaxFAR,
247              MatchResult  => MatchResult,
248              AchievedFAR  => AchievedFAR,
249              BioReturn    => NumericReturn);
250
251   if NumericReturn /= ValueOf(BioAPIOk) then
252     -- An error occurred, overwrite match
          information.
253     MatchResult := IandATypes.NoMatch;
254     AuditLog.AddElementToLog
255       (ElementID    => AuditTypes.SystemFault,
256        Severity     => AuditTypes.Warning,
257        User         => AuditTypes.NoUser,
258        Description  => MakeDescription
                           ("Biometric device failure ",
259                          NumericReturn));
260   end if;
261 end Verify;
```

If the log were accessible to potential hackers, which is not the case for Tokeneer, then this would be an example of an error message information leak. Fingerprint templates should never be accessible by hackers.

Step 2: Change an Information Flow Aspect of the Contract

Change the contract for the procedure Verify to specify that no information written to the log is derived from Template by deleting line 233.

```
230  --# derives AuditLog.State,
231  --#        AuditLog.FileState from AuditLog.State,
232  --#                            AuditLog.FileState,
233  --#
234  --#                            Clock.Now,
235  --#                            ConfigData.State,
236  --#                            Interface.Input &
```

The corresponding line (line 73) of code needs to be removed from the file bio.ads.

```
60   procedure Verify(Template      : in
                          IandATypes.TemplateT;
61               MaxFAR       : in   IandATypes.FarT;
62               MatchResult  :  out
                          IandATypes.MatchResultT;
63               AchievedFAR  :  out
                          IandATypes.FarT);
  ...
69   --# derives AuditLog.State,
70   --#        AuditLog.FileState from Input,
71   --#                            AuditLog.State,
72   --#                            AuditLog.FileState,
73   --#
74   --#                            Clock.Now,
75   --#                            ConfigData.State &
```

Step 3: Use the SPARK Tools to Detect the Information Leak

Examine the file bio.adb and notice the Examiner reports the error that information derived from the variable Template is written to the log. This means that data derived from the template is being written to the log!

```
bio.adb:261:8:
  Flow Error 601 - AuditLog.State may be derived from the
       imported value(s) of Template.
bio.adb:261:8:
  Flow Error 601 - AuditLog.FileState may be derived from
       the imported value(s) of Template.
```

Step 4: Introduce a Malicious Hack

The Examiner also detects when data has been incorrectly used. We now add back door code in the procedure Verify (lines 248-251 below). It returns a positive match independent of the user's fingerprint when the clock is at midnight.

```
223  procedure Verify(Template      : in
                          IandATypes.TemplateT;
224              MaxFAR       : in   IandATypes.FarT;
225              MatchResult  :  out
                          IandATypes.MatchResultT;
226              AchievedFAR  :  out
                          IandATypes.FarT)
  ...
232  --# derives AuditLog.State,
233  --#        AuditLog.FileState from AuditLog.State,
234  --#                            AuditLog.FileState,
235  --#                            Template,
236  --#                            Clock.Now,
237  --#                            ConfigData.State,
238  --#                            Interface.Input &
     ...
244  is
245    NumericReturn : BasicTypes.Unsigned32T;
246    T             : Clock.TimeT;
247  begin
248    T := Clock.GetNow;
249    if T = Clock.ZeroTime then
250      MatchResult := IandATypes.Match;
251      AchievedFAR := 0;
252    else
253
254      Interface.Verify(Template     => Template,
255                       MaxFAR       => MaxFAR,
256                       MatchResult  => MatchResult,
257                       AchievedFAR  => AchievedFAR,
258                       BioReturn    => NumericReturn);
259
     ...
272    end if;
     ...
275  end Verify;
```

Step 5: Use the SPARK Tools to Detect the Weakness

Examine the file bio_bad.adb and notice that the Examiner reports the error that MatchResult is dependent on the current time (Clock.now), which is inconsistent with the procedure's contract. The Examiner has identified an information flow inconsistency due to the presence of the back door.

```
bio_bad.adb:243:40:
  Flow Error   4 - The dependency of the exported value of
      AuditLog.State on the imported value
      of Verify.Template has not been previously stated.
bio_bad.adb:243:40:
  Flow Error   4 - The dependency of the exported value of
      AuditLog.FileState on the imported value
      of Verify.Template has not been previously stated.
bio_bad.adb:275:8:
  Flow Error 601 - MatchResult may be derived from the
      imported value(s) of Clock.Now.
bio_bad.adb:275:8:
  Flow Error 601 - AchievedFAR may be derived from the
      imported value(s) of Clock.Now.
```

Summary

In this Gem we have learnt about information flow contracts and we have seen the SPARK tools detect a malicious hack. SPARK programs are free from information leaks when the contract accurately specifies the desired information flow between variables.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o K.U. Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Peter Dencker
Steinäckerstr. 25
D-76275 Ettlingen-Spessartt
Germany
Email: dencker@web.de
URL: ada-deutschland.de

Ada-France

Ada-France
attn: J-P Rosen
115, avenue du Maine
75014 Paris
France
URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
Ada-Spain
P.O.Box 50.403
28080-Madrid
Spain
Phone: +34-942-201-394
Fax: +34-942-201-402
Email: gutierjj@unican.es
URL: www.adaspain.org

Ada in Sweden

Ada-Sweden
attn. Rei Stråhle
Rimbogatan 18
SE-753 24 Uppsala
Sweden
Phone: +46 73 253 7998
Email: rei@ada-sweden.org
URL: www.ada-sweden.org

Ada Switzerland

attn. Ahlan Marriott
White Elephant GmbH
Postfach 327
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: ada@white-elephant.ch
URL: www.ada-switzerland.ch