ADA USER JOURNAL

Volume 36
Number 2
June 2015

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

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

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

Proceedings and Special Issues

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

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

News and Product Announcements

Ada User Journal is one of the ways in which people find out what is going on in the Ada community. Our readers need not surf the web or news groups to find out what is going on in the Ada world and in the neighbouring and/or competing communities. We will reprint or report on items that may be of interest to them.

Reprinted Articles

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

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

All material for publication should be sent electronically. Authors are invited to contact the Editor-in-Chief by electronic mail to determine the best format for submission. The language of the journal is English.

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

Editorial

The issue of the Ada User Journal which you are reading is being finalized at the Ada-Europe 2015 conference, in Madrid, Spain, June 22-26. I would like to congratulate and thank the organizers for a very successful conference, with a rich program, and a pleasant social and networking atmosphere. In addition to the technical program, and taking the opportunity that the Ada community converged in Madrid, the conference celebrated the 200th anniversary of Lady Ada Byron Lovelace with the screening of the documentary “To Dream Tomorrow”, a film about Lady Ada, her work with Charles Babbage, and “their contributions to computing over a hundred years before the time usually thought to be the start of the Computer Age”.

As announced during the conference, Ada-Europe 2016 will take place in Pisa, Italy, in the week of 13-17 June, 2016. A great opportunity for Ada and Reliable Software practitioners and enthusiasts to present their work and for the community to connect in an enjoyable scenario. You can find the preliminary call for contributions in the Forthcoming Events section of this issue. Recognizing the importance of parallelism, and its impact on future reliable systems, the conference includes a Special Session on Safe, Predictable Parallel Software Technologies. Note that the program of the conference results from the contributions of the community, by means of the submission of papers, presentation, tutorials and workshops. I would like to both encourage, and insist in asking for, your contribution!

Also in the Events section, the Journal provides the announcement of the second UK conference on High Integrity Software which will take place in Bristol, UK, on November 5, 2015, an event about challenges and solutions in the domain of trustworthy software engineering for safety, security and business-critical applications. As usual the News Digest and Calendar sections, prepared by the respective Editors, Jacob Sparre Andersen and Dirk Craeynest, complete the first part of the issue.

The issue then continues the publication of articles related to the celebration of Ada Bicentennial, reprinting an article by John Fuegi and Jo Francis, directors of the “To Dream Tomorrow” documentary, originally published in the IEEE Annals of the History of Computing, October-December 2003, about the relation between Ada and Charles Babbage, and the creation of the Ada Lovelace's "Notes" describing the Analytical Engine.

As for the technical contents, the Journal continues the publication of contributions from the Ada-Europe 2015 conference with a paper from its industrial track, authored by a group of authors from Spazio IT, Italy, European Space Agency, The Netherlands, Inopus, Germany and AIRBUS Helicopters, Germany, on a code quality platform for the analysis of both Ada and C/C++ flight software.

Afterwards, we publish a document by authors from AdaCore and Altran UK, presenting how it is foreseen to support the Ravenscar profile in SPARK 2014. Concluding the issue, and continuing with SPARK, and the SPARK 2014 Rationale, the issue provides an article with contributions on Ghost Code, Object Oriented Programming and Functional Update by Yannick Moy of AdaCore, France.

Luís Miguel Pinho
Porto
June 2015
Email: AUJ_Editor@Ada-Europe.org

Quarterly News Digest

Jacob Sparre Andersen

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

Contents

Ada-related Organizations

Technical Guides - Update to Ada 2012?

From: Joyce L Tokar <tokar@pyrrhusoft.com>
Date: Mon, 27 Apr 2015 13:01:53 -0700
Subject: Technical Guides for the use of the Ada in high integrity systems
Newsgroups: comp.lang.ada

There two technical reports of the use of Ada in High Integrity Systems:

- ISO/IEC TR 15942:2000, Guidance for the Use of Ada in High Integrity Systems: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29575
- ISO/IEC TR 24718:2005, Guide for the use of the Ada Ravenscar Profile in high integrity systems:http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38828

The purpose of this discussion topic is to ask you, as members of the Ada community, if you are using these Technical Reports. And if you are interested in seeing these reports updated to be in alignment with Ada 2012?

Please response to me at tokar@pyrrhusoft.com

Thank You

Joyce L Tokar, PhD

Pyrrhus Software, LLC

ISO/IEC JTC 1/SC 22/WG 9 Convenor

Ada-related Events

[To give an idea about the many Ada-related events organised by local groups, some information is included here. If you are organising 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. —sparre]

Mascot Competition Result

From: David Botton <david@botton.com>
Date: Fri, 27 Mar 2015 12:37:55 -0700
Subject: The winner of the Ada Mascot Competition is.....
Newsgroups: comp.lang.ada

I am happy to announce the winner of the Ada Mascot Competition - Entry #5 - "Lady Fairy" the hummingbird. The mascot was designed by Leah Goodreau.

http://gnoga.com/mascot.html

Leah writes "My mascot was inspired by Ada Lovelace's adolescent fascination with flight and Charles Babbage's nickname for her, "Lady Fairy." The coloration of the mascot references the Asian fairy-bluebird, while the silhouette is a hummingbird because of their famed speed and sleekness."

If have asked Leah for permission to give out her contact information if any one is interested in custom versions of her work, any one interested in receiving her information, please feel free to contact me.

From: David Botton <david@botton.com>
Date: Mon, 30 Mar 2015 10:12:38 -0700
Subject: Ada Mascot Paraphernalia
Newsgroups: comp.lang.ada

Get your Ada Mascot Paraphernalia at:

http://www.cafepress.com/adamascot

I set the prophet level to zero, so lowest price they offer it.

Contest: Do Something Awesome with Ada

From: David Botton <david@botton.com>
Date: Tue, 31 Mar 2015 10:20:23 -0700
Subject: The LearnAdaNow.com Contest - Do Something Awesome with Ada
Newsgroups: comp.lang.ada

The LearnAdaNow.com Contest 2015

Prize @ $100 and Growing

Rules:

Create a recorded video (screen cast, talking head, full live action, etc.) of up to 1 hour long (no minimum) in mp4 format of

How to do something Awesome using Ada

1. Submit your entry to david@botton.com - if too large please provide a link for downloading.

2. The judges will be the prize donors and one representative from SIGAda and one from Ada Europe.

3. No one that submits an entry nor David Botton (the organiser) can be a judge.

4. Each judge will score each entry 1-10 points and the highest scored submission wins.

5. If you add a PDF transcript with static images of your recording and code samples if appropriate you automatically get 1 point added.

6. Each entry will be posted when received to LearnAdaNow.com

7. The last date for submissions will be July 31, 2015

8. The winner will be announced by August 7, 2015

9. All submitted videos and associated materials must have no restrictions on reuse and distribution and be original new works.

10. If there are multiple entries with the same highest score the winnings will be divided equally.

11. Multiple entries are allowed.

Prize Donors so Far:

David Botton - $100

To donate to the prize contact david@botton.com

Ada-Europe 2015 in Madrid

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Sun, 5 Apr 2015 09:47:41 +0000
Subject: 20th Int.Conf. Reliable Software Technologies, Ada-Europe 2015
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

Call for Participation

*** PROGRAM SUMMARY ***

20th International Conference on Reliable Software Technologies - Ada-Europe 2015

22-26 June 2015, Madrid, Spain

http://www.ada-europe.org/conference2015

Organised by Ada-Spain on behalf of Ada-Europe, in cooperation with ACM SIGAda, SIGBED, SIGPLAN and the Ada Resource Association (ARA)

*** Online registration open! ***

All info available on conference web site

Early registration discount until June 7

The 20th International Conference on Reliable Software Technologies - Ada-Europe 2015 takes place in Madrid, Spain, from June 22 to 26, 2015. It is an exciting event with an outstanding technical program, keynote talks, exhibition and networking from Tuesday to Thursday, and a rich program of workshops and tutorials on Monday and Friday.

The conference is hosted by ETSIT-UPM, the engineering school of the Polytechnic University of Madrid, which covers teaching and research in all fields related to Information and Communications Technology, and is one of the leading institutions in that field in Spain.

The Ada-Europe series of conferences has become established as a successful international forum for providers, practitioners and researchers in all aspects of reliable software technologies. These events highlight the increased relevance of Ada in safety- and security-critical systems, and provide a unique opportunity for interaction and collaboration between academics and industrial practitioners.

Extensive information is available on the conference web site, such as the list of accepted papers and industrial presentations, and detailed descriptions of all workshops, tutorials and keynote presentations. Also check the conference web site for registration, accommodation and travel information.

Quick overview

- Mon 22 & Fri 26: tutorials + workshops
- Tue 23 - Thu 25: core program

Proceedings

- published by Springer
- volume 9111 in Lecture Notes in Computer Science series (LNCS)
- will be available at conference

Program co-Chairs

- Juan A. de la Puente, Universidad Politécnica de Madrid, Spain

  jpuente@dit.upm.es
- Tullio Vardanega, Università di Padova, Italy

  tullio.vardanega@unipd.it

Invited speakers

- Jon Pérez, "EC-61508 Certification of Mixed-Criticality Systems based on Multicore and Partitioning"
- Javier Rodríguez, "Software Development of Safety-Critical Railway Systems"
- Andras Balazs, "The Central On-Board Computer of the Philae Lander in the Context of the Rosetta Space Mission"

Workshops (full day)

- Workshop on "Challenges and new Approaches for Dependable and Cyber-Physical Systems Engineering" (De-CPS 2015)
- Workshop on "Architecture Centric Virtual Integration" (ACVI 2015)

Tutorials (full day)

- "Parallelism in Ada, Today and Tomorrow", Brad Moore, General Dynamics Canada, and Stephen Michell, Maurya Software, Canada
- "Probabilistic Timing Analysis", Francisco J. Cazorla and Jaume Abella, Barcelona Supercomputing Center, Spain, Tullio Vardanega, University of Padua, Italy, and Mark Pearce, Rapita Systems Ltd, UK
- "Real-Time and Embedded Programming with Ada 2012", Patrick Rogers, AdaCore, USA

Tutorials (half day)

- "Access Types and Memory Management in Ada 2012", Jean-Pierre Rosen, Adalog, France
- "Designing and Checking Coding Standards for Ada", Jean-Pierre Rosen, Adalog, France
- "Ada 2012 (Sub)type and Subprogram Contracts in Practice", Jacob Sparre Andersen, JSA Research & Innovation, Denmark
- "When Ada meets Python: Extensibility through Scripting", Emmanuel Briot and Ben Brosgol, AdaCore, France and USA
- "Software Measures for Dependable Software Systems", William Bail, The MITRE Corporation, USA
- "Software Design Concepts and Pitfalls", William Bail, The MITRE Corporation, USA

Papers and Presentations

- 12 refereed technical papers in sessions on Language Technology, Real-Time Applications, Critical Systems, Multi-core and Distributed Systems
- 9 industrial presentations in sessions on Ada Applications, Critical Systems, Tools at Work
- 3 presentations in special "Advances on Methods" session

Vendor exhibition and networking area

- area features exhibitor booths, project posters, reserved vendor tables, and general networking options
- 3 companies already committed: AdaCore, Ellidiss Software, and Rapita Systems; others expected to confirm soon
- vendor presentation sessions in core program

Social events

- each day: coffee breaks in the exhibition space and sit-down lunches offer ample time for interaction and networking
- Tuesday evening: Welcome Cocktail
- Wednesday evening: the traditional Ada-Europe Conference Banquet will be held at Club de Campo Villa de Madrid, a country club located at the outskirts of the city, with magnificent views
- Ada Lovelace 200th Anniversary Celebration

Registration

- early registration discount up to Sunday June 7, 2015
- additional discount for academia, Ada-Europe, ACM SIGAda, SIGBED and SIGPLAN members
- a limited number of student discounts is available
- registration includes copy of printed proceedings at event
- includes coffee breaks and lunches
- three day conference registration includes all social events
- payment possible by credit card, check, or bank transfer
- see registration page for info on novel student waiver program!

Please make sure you book accommodation as soon as possible. Madrid will be very busy in that week.

For more info and latest updates see the conference web site at

http://www.ada-europe.org/conference2015.

Webinar: Security in Unmanned Aircraft Systems

From: Jamie Ayre <ayre@adacore.com>
Date: Mon Apr 13 2015
Subject: Upcoming webinar: Addressing security in safety-critical and mission-critical UAS
URL: http://blog.adacore.com/upcoming-webinar-addressing-security-in-safety-critical-and-mission-critical-uas

When it comes to unmanned aircraft systems (UAS), virtually everyone is talking about and concerned with privacy issues – as though drones were robotic peeping Toms. The much larger and more critical issue, however, is security – without it, the potential exists for control of drones and even swarms of drones to be usurped and used to inflict harm. UAS hardware and software must be designed with development tools proven to be effective in the design and deployment of

safety-critical and mission-critical systems and vehicles. In this webinar Robert Dewar will discuss the selection of optimal development tools and processes to ensure the safety, security, and reliability of real-time unmanned aircraft, onboard software, and ground control solutions.

Photographs from Ada-Europe 2014

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 29 Apr 2015 15:10:58 +0200
Subject: Photographs from Ada-Europe 2014/Paris

We finally managed to make a photo gallery with pictures from Ada-Europe 2014 in Paris:

http://www.adalog.fr/ae2014/gallery/

Enjoy! And I hope to see you in Madrid.

Ada-related Resources

Content for LearnAdaNow.com?

From: David Botton <david@botton.com>
Date: Wed, 1 Apr 2015 14:15:22 -0700
Subject: Non-Contest content for LearnAdaNow.com
Newsgroups: comp.lang.ada

While the contest is to give the Ada community video content that will show Awesome stuff you can do with Ada. The LearnAdaNow.com site is intended to be a vehicle for advocating Ada in general to those _not_ in the Ada community already, once built up I will make sure the entire software world sees it :)

If you have articles, tutorials, etc. for anything cool with Ada, please e-mail them to me. I'll start collecting what I can in general, but any help to build it up the better.

Ada Information Clearinghouse

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed Apr 22 CEST 2015
Subject: New home page launched
URL: http://www.adaic.org/2015/04/new-home-page-launched/

We've update our home page to better reflect our mission. Our old home page over-emphasized Ada news and said nothing about the many resources that we have for users of the Ada programming language (including existing, new, and potential users). The highlighted resources will be changed periodically to show different facets of our site.

The new home page has a simplified version of the Ada news feed on the left side. If, however, you prefer the old, more detailed news feed, it can be found on the news page[1].

As always, feel free to send us feedback on the new home page or anything else via our contact page[2].

[1] http://www.adaic.org/news/

[2] http://www.adaic.org/contact/

Ada on Social Media

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Fri Apr 24 2015
Subject: Ada on Social Media

Ada groups on various social media:

- LinkedIn[1]: 2_229 members
- Reddit[2]: 780 readers
- Google+[3]: 475 members
- StackOverflow[4]: 274 followers
- Twitter[5]: 4 tweeters

[1] http://www.linkedin.com/groups?gid=114211

[2] http://www.reddit.com/r/ada/

[3] https://plus.google.com/communities/102688015980369378804

[4] http://stackoverflow.com/questions/tagged/ada

[5] https://twitter.com/search?f=realtime&q=%23AdaProgramming

[See also "Ada on Social Media", AUJ 36-1, p. 10. —sparre]

Open Source Build Server Status

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Fri Apr 24 2015
Subject: Jenkins
URL: http://build.ada-language.com/

[Builds: —sparre]

- Ahven - Debian 7.0 - GNAT 4.6
- Ahven_JNT
- Ahven_Win7_GNAT2013
- Ahven_Win7_ICCAda
- JD_JNT
- Jdaughter - Debian 7.0 - GNAT 4.6
- Jdaughter_Win7_ICCAda
- Lace_Win7_ICCAda

[Fails to build: —sparre]

- AVR-Ada_Debian_7
- Strings_Edit_ICCAda
- UnzipAda_Win7_GNAT2013
- UnzipAda_Win7_ICCAda

[See also "Open Source Build Server Status", AUJ 36-1, p. 10. —sparre]

Repositories of Open Source Software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Fri May 1 2015
Subject: Repositories of Open Source software

GitHub: 842 repositories [1]
233 developers [1]

Rosetta Code: 616 examples [2]
29 developers [3]

Sourceforge: 237 repositories [4]

BlackDuck OpenHUB: 210 projects [5]

Bitbucket: 110 repositories [6]
17 developers [6]

OpenDO Forge: 24 projects [7]
431 developers [7]

Codelabs: 20+ repositories [8]

AdaForge: 8 repositories [9]

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

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

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

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

[5] https://www.openhub.net/tags/ada

[6] http://edb.jacob-sparre.dk/Ada/on_bitbucket

[7] https://forge.open-do.org/

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

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

[See also "Repositories of Open Source Software", AUJ 36-1, p. 10. —sparre]

Ada-related Tools

Socket Libraries

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 26 Dec 2014 19:21:48 -0600
Subject: Re: Questions about socket programming
Newsgroups: comp.lang.ada

> [...] GNAT.Sockets [...]

Unless you want it to work with other Ada compilers (to avoid GNAT lock-in).

> [...]

My complaint is simply that you end up with GNAT lock-in whenever you depend on GNAT-specific packages. Ada is powerful enough that such lock-in shouldn't be necessary. So it's better to use a package that is more portable, like Claw.Sockets (if you're already locked into Windows) or AdaSockets or (coming soon) NC_Sockets.

[See also “AdaSockets”, AUJ 34-3, p. 141. —sparre]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 27 Dec 2014 10:48:36 +0100
Subject: Re: Questions about socket programming
Newsgroups: comp.lang.ada

[...]

Obviously sockets should be included into the standard library. Even considering embedded targets, socket I/O is probably more relevant there than text I/O, which is a part of the library.

GNAT sockets could be a good reference point. The only important (for embedded applications) part missing is raw sockets.

> [...]

Actually GNAT sockets are more portable than AdaSockets, as they work on a wider set of targets (e.g. VxWorks). You mean compiler independence.

Gnoga

From: David Botton <david@botton.com>
Date: Tue, 13 Jan 2015 17:36:30 -0800
Subject: Full direct SSL support now in Gnoga
Newsgroups: comp.lang.ada

Now in Gnoga full direct SSL support is now available (Special thanks to Dmitry for adding HTTPS support to simple components).

To add HTTPS support you simply add:

```
with "..path..to../ssl/gnoga_secure.gpr";
```

Then:

```
Gnoga.Server.Connection.
Secure.Register_Secure_Server
(Certificate_File =>  "path_to_certificate.crt",
     Key_File  =>      "path_to_keyfile.key",
    Port  => 8443,
    Disable_Insecure => False);

Gnoga.Application.Multi_Connect.Initialize
(Port => 8082);
```

Your server now listens to HTTP on 8082 and HTTPS on 8443.

Alternatively you cannot use direct support and use an SSL proxy (see the FAQ).

[See also “Gnoga”, AUJ 36-1, p. 12. —sparre]

Request: Binding to Amazon Simple Queue System

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Mon, 19 Jan 2015 18:02:14 +0100
Subject: Amazon SQS (simple Queue system) binding ?
Newsgroups: comp.lang.ada

I'm looking at interfacing to a set of Java programs via Amazon's Message Queues <http://aws.amazon.com/sqs/>

I wonder if anyone has written bindings for Ada for it, before I roll my own.

The choice of SQS was not mine, but a friend's who wants to explore the scalability of the thing.

It looks fairly simple/straightforward, but I thought I'd ask anyway.

Ada to JavaScript Translator

From: Tom Moran <tmoran@acm.org>
Date: Tue, 20 Jan 2015 20:43:51 +0000
Subject: Ada->Javascript?
Newsgroups: comp.lang.ada

Does there exist any kind of Ada to JavaScript translator? Even something that took compilable, but not runnable, simple Ada to JavaScript. Having an Ada compiler check for typos, type errors, etc, and handle enumeration types, non-zero array'first, etc etc would be a great help.

From: Vadim Godunko <vgodunko@gmail.com>
Date: Wed, 21 Jan 2015 03:25:06 -0800
Subject: Re: Ada->Javascript?
Newsgroups: comp.lang.ada

> [...]

There is ongoing work, see http://forge.ada-ru.org/matreshka/wiki/Web/AdaToJavaScript/Examples.
It is very limited now, features requests are welcome.

GNAT: An Optimisation Flag

From: Simon Wright <simon@pushface.org>
Date: Sat, 14 Feb 2015 17:40:53 +0000
Subject: Optimisation
Newsgroups: comp.lang.ada

With GCC (and GNAT), there's an optimisation level I hadn't come across before[1]:

-Og

Optimize debugging experience.

Quite a tall order!

[1] https://gcc.gnu.org/onlinedocs/gcc-4.9.2/gcc/Optimize-Options.html

Gprbuild: Improvements in Code Generation Support

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Sun Mar 15 2015
Subject: Gem #157: Gprbuild and Code Generation
URL: http://www.adacore.com/adaanswers/gems/gem-157-gprbuild-and-code-generation/

Do you have any plans to make it possible to invoke `gprbuild' only once, instead of once for each compiler used by the project?

From: Emmanuel Briot <briot@adacore.com>
Date: Mon May 4 2015
Subject: Gem #157: Gprbuild and Code Generation
URL: http://www.adacore.com/adaanswers/gems/gem-157-gprbuild-and-code-generation/

As a matter of fact, we have indeed made good progress on such a design.

I think your question is slightly misleading: of course, if each of the compiler is associated with a different language, and there is no time-order dependency between those languages, gprbuild is happy to process with the builds of all languages in parallel.

The issue, of course, is when one of the languages (say for code generation), is used to generate sources for another language (as in this gem). In this case, the current solution is to spawn multiple gprbuilds (and is really and adequate solution in practice). We will likely go towards a solution similar to the scenario variables (but the value of the variable would be set automatically by gprbuild to indicate the build phase). Gprbuild will then build all languages associated with the first build phase, then all languages for the second build phase, and so on. That means that a single gprbuild command will be enough. Unfortunately, we have so far not found a scheme where gprbuild would be able to automatically build in parallel (as much as possible) the various phases. So in practice the effect will be very similar to spawning multiple gprbuilds one after the other, and use the -X command line switch.

Zip-Ada

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue, 24 Mar 2015 11:04:19 -0700
Subject: Ann: Zip-Ada v.49
Newsgroups: comp.lang.ada

There is a new version of Zip-Ada @ unzip-ada.sf.net . Zip-Ada is a library for dealing with the Zip compressed archive file format. It supplies:

- compression with the following sub-formats ("methods"): Store, Reduce, Shrink (LZW) and Deflate
- decompression for the following sub-formats ("methods"): Store, Reduce, Shrink (LZW), Implode, Deflate, BZip2 and LZMA
- encryption and decryption (portable Zip 2.0 encryption scheme)
- unconditional portability (see below a list of in-use platforms)
- input (archive to decompress or data to compress) can be any data stream
- output (archive to build or data to extract) can be any data stream

- Zip_info and Zip_Create_info to handle quickly and easily archives

- cross format compatibility with the most various tools and file formats based on the Zip format: 7-zip, Info-Zip's Zip, WinZip, PKZip, Java's JARs, OpenDocument files, MS Office 2007+, Nokia themes, and many others

- task safety: this library can be used ad libitum in parallel processing

- endian-neutral I/O

Latest changes since v.47

* Changes in '49', 21-Mar-2015:

- encryption implemented (portable Zip 2.0 encryption scheme)

* Changes in '48', 20-Jul-2014:

- LZMA decompression significantly faster

* Changes in '47', 28-Jun-2014:

- LZMA method added for decompression

[See also “Zip-Ada”, AUJ 35-3, p. 157. —sparre]

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue, 24 Mar 2015 14:05:13 -0700
Subject: Re: Ann: Zip-Ada v.49
Newsgroups: comp.lang.ada

> Perhaps someone could make a project of converting AWS to use your zip code instead of the current C libs it uses.

I guess: you mean replacing zlib by an Ada solution - and perhaps some headaches with .dll's, versions and so on?

There is a "placeholder" project for that (Zada at SourceForge) so the question is to rip the Deflate compression and decompression from Zip-Ada and mix it with the body of zlib-Ada. If someone familiar with zlib and zlib-Ada volunteers to help with it, I can plug the compression and decompression code at the right place (I hope). I miss the experience with zlib itself.

STM32F4 GNAT Run Time Systems

From: Simon Wright <simon@pushface.org>
Date: Mon, 06 Apr 2015 18:27:09 +0100
Subject: ANN: STM32F4 GNAT Run Time Systems 20150406
Newsgroups: comp.lang.ada

This is the fourth release of a GNAT RTS with the GCC Runtime Library exception for STM32F4 boards.

(a) Tasking is implemented using FreeRTOS[3], which STMicroelectronics provide a copy of with their BSP.

(b) I've included a BSP with minimal higher-level Ada interfaces to the board hardware: clock, buttons, LEDs, LCD. In addition, there's a machine-generated translation of STMicroelectronics' type-specific header in stm32f429xx_h.ads, for low-level interfacing.

The release is at https://sourceforge.net/projects/stm32f4-gnat-rts/files/20150406/.

This release has been reorganised from previous releases.

There is one RTS, stm32f4-disco-rtos, and one BSP, stm32f4-disco-bsp.

Changes to the RTS from the previous release:

These units (and supporting units) are now included:

Ada.Containers.Bounded_Vectors (*)

Ada.Containers.Bounded_Hashed_Maps (*)

Ada.Containers.Generic_Array_Sort

Ada.Containers.Generic_Constrained_Array_Sort

Ada.IO_Exceptions

Ada.Streams

Ada.Task_Identification

Interfaces.C

Interfaces.C.Strings

System.Assertions

(*) The new iterators (for some F in Foo loop ...) are NOT supported (they require finalisation).

The STM32F429I_Discovery tree has been moved to the BSP.

The following tickets have been fixed:

2 Protected spec hides package Interfaces

14 Last_Chance_Handler doesn’t stop tasking

Tasking is started by calling Start_FreeRTOS_Scheduler.

[See also “STM32F4 GNAT Run Time Systems”, AUJ 36-1, p. 15. —sparre]

Stream Tools

From: Per Sandberg <per.s.sandberg@bahnhof.se>
Date: Wed, 08 Apr 2015 08:42:27 +0200
Subject: ANN: stream_tools 1.0.1
Newsgroups: comp.lang.ada

First release of stream-tools

https://github.com/persan/a-stream-tools/releases/tag/1.0.1

They provides a small set of utility streams.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 8 Apr 2015 16:41:45 -0500
Subject: Re: stream_tools 1.0.1
Newsgroups: comp.lang.ada

> [...]

For what it's worth, there was/is a proposal to include what you called a "memory stream" in Ada 202x. There's been one in Claw since it was created, and I hear that it comes up frequently enough that it probably should be standard.

The Claw version (Claw.Marshalling.Buffer_Type) uses a discriminated type so that the dangerous use of 'Address isn't necessary. It expands the buffer when necessary, but I would expect that would get dropped from a Standard version. And probably the names would get changed. But here's the spec:

```
package Claw.Marshalling is
  type Buffer_Type (Initial_Length :
    Ada.Streams.Stream_Element_Count)
    is new Ada.Streams.Root_Stream_Type
   with private;

  procedure Read
    (Stream : in out Buffer_Type;
     Item   :    out
          Ada.Streams.Stream_Element_Array;
     Last   :    out
        Ada.Streams.Stream_Element_Offset);

  procedure Write
    (Stream : in out Buffer_Type;
     Item   : in
          Ada.Streams.Stream_Element_Array);

  function Length (Stream : in Buffer_Type)
    return
        Ada.Streams.Stream_Element_Count;
  -- Return the total length of data
  -- written into the buffer.

  procedure Clear (Stream : in out
            Buffer_Type);
  -- Empty the buffer.
private
  ...
```

Emacs Ada Mode

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Wed, 15 Apr 2015 03:10:42 -0500
Subject: Emacs ada-mode 5.1.8 released
Newsgroups: comp.lang.ada

Emacs ada-mode 5.1.8 is available on the website (http://stephe-leake.org/emacs/ada-mode/emacs-ada-mode.html) and in Gnu ELPA.

See http://stephe-leake.org/emacs/ada-mode/NEWS-ada-mode.text for major changes.

One backwards incompatibility: ada-case-identifier now takes three args; this allows capitalising more sensibly in strings and comments. So if you have that set to 'upcase-region anywhere, you need to change it to 'ada-upper-case.

Otherwise this is a bug fix release.

This requires the new OpenToken 6.0 (http://stephe-leake.org/ada/opentoken.html) if building from source.

[See also “Emacs Ada Mode”, AUJ 36-1, p. 15. —sparre]

OpenToken

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Wed, 15 Apr 2015 03:09:34 -0500
Subject: OpenToken 6.0 released
Newsgroups: comp.lang.ada

OpenToken 6.0 is now available on the website: http://stephe-leake.org/ada/opentoken.html

See http://stephe-leake.org/ada/opentoken.html#History for a description of the changes.

The main change is support of generalised LALR (spawn parallel parsers) to handle conflicts in the Ada parser runtime (previous releases only supported this in the parse table generator; the runtime was provided only in Emacs elisp).

The API of the OpenToken packages changes significantly, due to reorganising/cleaning up to support generalised LALR. So existing projects will have to be edited. The changes are only in the instantiations, not in user code logic.

[See also “OpenToken”, AUJ 35-1, p. 10. —sparre]

ColdFrame

From: Simon Wright <simon@pushface.org>
Date: Thu, 16 Apr 2015 17:43:19 +0100
Subject: ANN: ColdFrame 20150415
Newsgroups: comp.lang.ada

This announces release 20150415 of ColdFrame, which generates Ada code frameworks from UML models in ArgoUML.

Changes from previous releases can be seen at the Files link, but:

* Bounded containers are used where possible.

* If required, Ravenscar-compliant code can be generated.

* Includes support for scripted testing.

Project: https://sourceforge.net/projects/coldframe/

Web: http://coldframe.sourceforge.net/

Files: https://sourceforge.net/projects/coldframe/files/coldframe/20150415/

[See also “ColdFrame UML to Ada translator”, AUJ 33-2, p. 80. —sparre]

Mosquitto

From: Per Sandberg <per.s.sandberg@bahnhof.se>
Date: Mon, 20 Apr 2015 07:12:33 +0200
Subject: ANN:mosquitto-ada 0.0.1
Newsgroups: comp.lang.ada

A binding to the MSQTT broker mosquitto. It is a complete initial binding to the transport.

https://github.com/persan/mosquitto-ada

ASIS Components

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 21 Apr 2015 10:37:23 +0200
Subject: ASIS users: Scope manager added to Adalog's ASIS components
Newsgroups: comp.lang.ada

Talking about managing scopes...

I've added the scope manager to Adalog's ASIS components. It is a component that has been used for a long time in AdaControl (and therefore extensively tested), but I made it independent from AdaControl for others to reuse. GMGPL of course.

Get it from: http://sourceforge.net/p/adacontroladalog-asiscomps/

It depends on Thick_Queries (available from the same place) and Binary_Map (available from Adalog general components).

PCAB

From: Ali Bendriss <ali.bendriss@gmail.com>
Date: Tue, 28 Apr 2015 16:54:59 +0000
Subject: ANN PCAB 0.1
Newsgroups: comp.lang.ada

I have put online PCAB, an Ada binding to libpcap.

The Ada thin layer is generated using the gcc binding generator. There is as well a thick wrapper (pcap.ads) to hide most of the low level stuff.

You may find more info + a download link by following this URL:

http://wiki.tele-solve.com/PacketCaptureAdaBinding

Please let me know if you have any idea of improvement.

Generic Image Decoder

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue May 5 2015
Subject: Gautier's blog: GID release #04
URL: http://gautiersblog.blogspot.dk/2015/05/gid-release-04.html

GID means Generic Image Decoder, an open-source library that can be found here:

http://gen-img-dec.sourceforge.net/

In the latest version, in addition to the sophisticated formats like JPEG and PNG, the decoder supports also the simple format family PNM (Portable aNy Map) with the flavors PBM (Portable BitMap, black & white), PGM (Portable GreyMap), PPM (Portable PixMap, in full colors).

[...]

[See also “Generic Image Decoder”, AUJ 33-4, p. 236. —sparre]

Ada-related Products

GNAT Pro

From: AdaCore Press Center
Date: Tue Feb 24 2015
Subject: AdaCore Releases GNAT Pro 7.3
URL: http://www.adacore.com/press/gnatpro7-3/

New version of Ada Development Environment highlights annual major release of company product line

EMBEDDED WORLD 2015, Nuremberg, Germany, February 24, 2015 – AdaCore, the leading provider of commercial software solutions for the Ada programming language, today released GNAT Pro 7.3, the latest version of the company’s flagship Ada Development Environment. GNAT Pro 7.3 incorporates performance improvements, new functionality, and many other enhancements. It is part of the annual cycle of a major release for the company’s products, and Q1 2015 will also see new versions of the CodePeer deep static analysis tool for Ada and the SPARK Pro verification environment for high-integrity software, as well as the launch of the QGen model-based development and verification tool for Simulink® and Stateflow® models.

GNAT Pro includes a full Ada compiler, Integrated Development Environments – the GNAT Programming Studio (GPS) and the Eclipse-based GNATbench – a comprehensive toolset including a visual debugger, and an extensive set of libraries and bindings.

GNAT Pro 7.3 incorporates upgraded technology for the back end (GCC 4.9) and debugger (GDB 7.8) and includes more than 175 new features, many of which are based on customer recommendations. Enhancements include the following:

- improved diagnostic messages

- fine-grained control over the treatment of warnings

- extended support for non-default endianness

- a math library on bare-board platforms, designed for use in safety-certified systems

- support for large files on 32-bit systems

- improved handling of inlining

- overflow checks enabled by default

- enhanced code generation and debugging capabilities

In addition, most GNAT Pro tools now support aggregate projects. For more efficient performance a number of tools, including GNAT2XML and GNATmetric , can take advantage of parallel and incremental processing, and GNATtest now supports the stubbing of units.

”For more than 15 years now, new versions of GNAT Pro have been released annually according to schedule and at a level of quality required for building and maintaining mission- and safety-critical systems” said Cyrille Comar, President of AdaCore. “The challenge then and now is to provide the right balance between a high level of stability and the constant need for evolution and innovation. The latest version of GNAT Pro shows that we have met this challenge, allowing our customers to actively maintain their long-lived systems without being hampered by obsolete techniques and technology.”

[See also “GNAT Pro”, AUJ 35-2, p. 81. —sparre]

QGen

From: AdaCore Press Center
Date: Tue Feb 24 2015
Subject: AdaCore Launches QGen
URL: http://www.adacore.com/press/adacore-launches-qgen/

Customisable code generator and model verifier for Simulink and Stateflow models is designed for qualification against software safety certification standards.

Nuremberg – Embedded World Conference, NEW YORK and PARIS, February 24, 2015 – AdaCore today announced the release of QGen 1.0, a qualifiable and customisable code generator and model verifier for Simulink and Stateflow models. This tool can generate MISRA C and SPARK source code producing readable, traceable, and efficient code. It is particularly suited for developing and verifying high-integrity real-time control applications, especially where safety certification is required. The tool is highly configurable thanks to its visible intermediate representation.

QGen handles around 100 Simulink blocks. These were selected as a safe subset that guarantees predictable code generation patterns, does not require any run-time support, and allows for tool qualification against software safety standards. Support for Stateflow models is expected during late Q2 2015.

The tool's static model verifier detects run-time errors such as integer overflow and division by zero. It also can find logic errors such as dead execution paths, and verify functional properties through Simulink Assertion blocks. QGen can be integrated with AdaCore’s GNATemulator and GNATcoverage tools to support Processor-in-the-Loop (PIL) testing and structural coverage analysis without any code instrumentation.

Qualification material for QGen will be available for standards such as DO-178C (avionics), EN 50128 (rail), and ISO 26262 TCL3 (automotive). The model verification feature is qualifiable for DO-178C at Tool Qualification Level 5.

A QGen demo is available at http://www.adacore.com/qgen_demo.

“Thanks to its strong focus on safety, QGen reinforces the position of the Simulink and Stateflow environments as the preferred solutions for model-driven development of high-integrity control systems,” said Matteo Bordin, product manager for QGen at AdaCore. “With QGen, AdaCore offers a uniquely integrated and qualifiable solution for end-to-end model-based design, including code generation, production of high-performance embedded code, formal verification, structural coverage, and support for Processor-In-The-Loop testing.

“QGen offers us two opportunities,” said Cyrille Comar, AdaCore President. “First, our existing customers can now benefit from code generation from Simulink/Stateflow models in a way that is compatible and integrated with their existing tool and language investment. Further, QGen’s outstanding capabilities are attracting interest from new application domains driven by safety-critical requirements; this allows us to provide our high-integrity expertise and toolset to a much larger user base.”

VectorCAST

From: Vector Software
Date: Tue Feb 24 2015
Subject: Vector Software Releases VectorCAST 6.3
URL: https://www.vectorcast.com/news/vector-software-press-releases/2015/vector-software-releases-vectorcast-63

Smaller Footprint and Safety-Critical Expertise Delivers an loT / M2M - Ready Test Environment

Vector Software, the world’s leading provider of innovative software solutions for robust embedded software quality, announced the release of VectorCAST™ 6.3 today, the most Internet of Things (IoT) and Machine-to-machine (M2M)-ready embedded test suite.

Building on the embedded domain expertise Vector Software has developed over the last 20 years, version 6.3 provides a new micro harness architecture designed for the special needs of IoT / M2M applications. The new architecture is critical for IoT / M2M applications because of the smaller microprocessors and limited resources that are available to these applications. Analysts are projecting IoT and M2M to grow into a $71 billion industry by 2018 (Juniper Research, Smart Home Ecosystems & the Internet of Things, 12/02/2014). With billions of newly connected devices going on line, high quality software is essential for these devices to operate as intended.

“The coming growth of IoT and M2M will take a slice of intelligence out of the cloud and push it back to the periphery of the network,” said John Paliotta, Chief Technology Officer, Vector Software. “As this evolves, the correct autonomous operation of those network end points will be critical. We feel that VectorCAST is uniquely positioned to help developers build quality into IoT and M2M applications.”

Beyond the new harness architecture, VectorCAST 6.3 provides several other enhancements for test collaboration, Change-Based Testing (CBT), and massively parallel testing. These features make it simple for your team to leverage test cases and test results across the enterprise. Each developer can quickly and independently test their code changes by running only those tests affected by the source code changes made; this leads to improved software quality with reduced test cycle times.

As tests are integrated together into suites, an enhanced integration with the open-source continuous integration server Jenkins, provides VectorCAST 6.3 users the ability to deploy massively parallel testing over hundreds of servers.

To learn more about VectorCAST 6.3, please visit us here: https://www.vectorcast.com/vectorcast-63

To get the VectorCAST 6.3 release notes, please visit us here: https://www.vectorcast.com/downloads

[...]

Rapita Verification Suite

From: Rapita Systems
Date: Mon Mar 23 2015
Subject: First Code Coverage Solution for Multi-Core Systems Announced by Rapita Systems
URL: http://www.rapitasystems.com/news/first-code-coverage-solution-multi-core-systems-announced-rapita-systems

Rapita Systems Ltd, leading provider of on-target verification solutions for critical, real-time embedded systems, is pleased to announce the availability of version 3.3 of Rapita Verification Suite (RVS). This release features a range of enhancements including the first multi-core code coverage solution.

RVS is Rapita Systems' solution for supporting on-target verification for critical, real-time, embedded systems in industries such as avionics and automotive. It features tools to track structural code coverage (RapiCover),

measure timing behavior and to predict worst-case execution time (RapiTime), and to track system behavior within real-time operating systems and/or across multiple processor cores.

The latest v3.3 release introduces a range of features with a particular benefit for users performing structural coverage analysis. Multi-core code coverage allows a user to identify which cores executed specific code during testing. RapiCover's extremely low overheads, which already have a big impact in reducing the number of repetitions of tests are reduced to even lower levels with this new version. Effort to certify, which is a major consideration for customers using coverage tools with DO-178C or ISO 26262, is further reduced in RVS 3.3 through several features, such as justifications, which allow non-executed code to be highlighted and explained.

Rapita Systems CEO, Dr Guillem Bernat commented "In RVS 3.3 our aim is to provide the best structural coverage analysis tool for critical embedded systems."

"Measuring code coverage of tests has always played a major part in the verification of critical systems and we are increasingly seeing a demand for on-target code coverage. This puts a strong emphasis on tools that minimize the impact of measuring coverage. RapiCover's industry-leading low overheads have already resulted in massive reductions in testing effort for our customers. In RVS 3.3, we reduce the overheads of RapiCover still further. "

Bernat continued "Looking forward, we see increasing numbers of customers adopting multi-core processors. For some customers it won't be sufficient just to say code ran on one of the cores – instead they will need to know that code ran on a specified sub-set of the cores, but definitely was not executed on others."

RVS 3.3 builds on Rapita's successful track record of bringing verification products to engineers working on critical real-time embedded systems in the avionics and automotive electronics industries.

[...]

Early Access to Advanced Verification Technologies

From: Rapita Systems
Date: Mon Apr 20 2015
Subject: Working with advanced technologies via Rapita Systems' Early Access Program
URL: http://www.rapitasystems.com/system/files/downloads/MC-PB-011-54%20LR%20Early%20Access%20Program.pdf

Introduction to the Early Access Program

Rapita Systems has always been heavily involved in the research and development of advanced verification techniques for high-integrity systems.

Recognizing that our customers can benefit from this research even before it is fully developed into a product, Rapita has defined the Early Access Program (EAP) as a way for customers to access these technologies.

Sometimes our customers have problems that cannot be solved with commercially available tools, although they could be addressed with technologies that Rapita Systems is developing internally. This technology comes from our involvement in collaborative EU research programs as well as our internal product development process. When a customer comes to us with a specific requirement that could be addressed by EAP technologies, we jointly devise a package of consultancy and tools derived from the EAP that is specifically tailored to the project's needs.

Types of Technology covered by the EAP

Within the EAP we focus on technologies closely related to Rapita's core mission of verification of high-reliability, embedded and real-time systems. This includes:

- Techniques for doing on-target verification, including timing, coverage, stack-usage, cache, and tracing.
- Technologies for capturing/logging data from the target, including capturing data in real-time and on-target.
- Technologies related to analyzing source code.
- Multicore and many-core.
- Automotive and Aerospace-specific standards, tools and technologies.

[...]

Ada and Operating Systems

Debian: GtkAda for ARMv7

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 2 Apr 2015 10:20:32 +0200
Subject: ANN. GtkAda 3.8.2 for ARM
Newsgroups: comp.lang.ada

I have packaged GtkAda as distributed with GNAT GPL 2014 for Debian ARMv7.

ARMv7, also known as armhf, is what you get with Raspberry Pi 1/2 and BeagleBone. Here is the link:

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

Note, this is not an official release, I am not Debian GtkAda maintainer.

Debian: Libraries Supported on ARMv7

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 2 Apr 2015 11:27:56 +0200
Subject: ANN: ARM support update
Newsgroups: comp.lang.ada

The following libraries are now supported on ARMv7 and packaged for Debian armhf.

Ada industrial control widget library

http://www.dmitry-kazakov.de/ada/aicwl.ht

[See also "Industrial Control Widget Library", AUJ 35-3, p. 157. —sparre]

Fuzzy sets for Ada

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

[See also "Fuzzy Sets", AUJ 35-3, p. 157. —sparre]

GtkAda contributions

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

[See also "GtkAda Contributions", AUJ 35-3, p. 155. —sparre]

Interval arithmetic

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

[See also "Interval Arithmetic", AUJ 35-3, p. 157. —sparre]

Measurement units for Ada

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

[See also "Units of Measurement", AUJ 35-3, p. 156. —sparre]

Simple components for Ada

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

[See also "Simple Components", AUJ 35-3, p. 154. —sparre]

String editing

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

[See also "Strings_Edit", AUJ 35-3, p. 154. —sparre]

Table management

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

[See also "Tables", AUJ 35-3, p. 154. —sparre]

GNAT programming studio (GPS) library installer

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

Fuzzy machine learning framework

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

[See also “Fuzzy machine learning framework”, AUJ 33-3, p. 143. —sparre]

Archlinux: Available Packages

From: Rod Kay <rodakay@internode.on.net>
Date: Sat, 18 Apr 2015 06:07:11 -0700
Subject: Updated Ada support for Archlinux.
Newsgroups: comp.lang.ada

I've added/updated a few of the Ada libs/tools for Archlinux. Hopefully, they will update Ada support to be equivalent to what is available on FreeBSD (whose ports were used as a basis for many of the Archlinux updates).

Here is a list of the packages available ...

- ada-web-server 3.2.0
- ahven 2.4
- asis gpl 2014
- florist gpl 2014
- gnat-gps 6.1.0
- gnat_util 4.9.2
- gprbuild gpl 2014
- gtkada 3.8.3.1
- polyorb gpl 2014
- xmlada gpl 2014

The existing 'gcc-ada' package provides support for gcc 4.9.2.

Any feedback via the usual AUR site would be appreciated.

If anyone can suggest other Ada related packages to add, please do.

Mac OS X: GCC

From: Simon Wright <simon@pushface.org>
Date: Thu, 30 Apr 2015 12:22:17 +0100
Subject: ANN: GCC 5.1.0 for Mac OS X
Newsgroups: comp.lang.ada

See https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/5.1.0/

This is GCC 5.1.0 built for Mac OS X Mavericks (10.9.5, Darwin 13.4.0), with the Command Line Tools for Xcode 6.2. It also runs on Yosemite.

Compilers included: Ada, C, C++, Objective C, Objective C++, Fortran.

Tools included:

Full GPL:

ASIS, AUnit, GDB, GNATColl, and GPRbuild from GNAT GPL 2014.

GPL with Runtime Library Exception[1]:

XMLAda from the public SVN repository[2] at revision 238235 (XMLAda-SVN for short).

The gory details at http://forward-in-code.blogspot.co.uk/2015/04/building-gcc-510.html

From: Simon Wright <simon@pushface.org>
Date: Thu, 30 Apr 2015 13:23:08 +0100
Subject: Re: ANN: GCC 5.1.0 for Mac OS X
Newsgroups: comp.lang.ada

> [...]

One user-visible change I've noted, in GNAT.Sockets, is that Vector_Element.Length (used in Vector_Type, used in Receive_Vector and Send_Vector) is now of type Interfaces.C.size_t; used to be Ada.Streams.Stream_Element_Count. I guess this is for efficiency in scatter-gather operations.

Mac OS X: GCC for ARM-EABI

From: Simon Wright <simon@pushface.org>
Date: Fri, 01 May 2015 16:24:05 +0100
Subject: ANN: GCC 5.1.0 arm-eabi for Mac OS X
Newsgroups: comp.lang.ada

See https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/5.1.0/

This is GCC 5.1.0, rebuilt as a cross-compiler from Mac OS X to arm-eabi (specifically, the Cortex-M4 as found on the STMicroelectronics[1] STM32F4 Discovery and STM32F429I Discovery boards).

The compiler comes with no Ada Runtime System (RTS). See the STM32F4 GNAT Run Time Systems project[2] for candidates.

For details, see [3].

[1] http://www.st.com

[2] https://sourceforge.net/projects/stm32f4-gnat-rts/

[3] http://forward-in-code.blogspot.co.uk/2015/05/building-gcc-510-for-arm-eabi.html

[See also “Mac OS X: GNAT GPL 2014 for ARM-EABI”, AUJ 36-1, p. 17. —sparre]

References to Publications

Driving Neopixel LEDs

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue Mar 17 00:00:00 CET 2015
Subject: Driving Neopixel LEDs using only Ada
URL: http://arduino.ada-language.com/driving-neopixel-leds-using-only-ada.html

Inspired by my earlier delay experiments and its follow-up discussion on AVR-Ada mailinglist, I decided to put my delay functions in good use.

Neopixel RGB LEDs require exact timing and people usually use AVR assembler code to get the timing right. However, I wanted to see can I do it with plain Ada on normal Arduino.

The short answer is: Yes, it is doable.

[...]

A Building Code for Building Code

From: Yannick Moy
Date: Wed Mar 25 2015
Subject: A Building Code for Building Code
URL: http://blog.adacore.com/a-building-code-for-building-code

If you can't make sense of the title of this post, you may need to read the recent article about it in Communications of the ACM[1]. In this article, Carl Landwehr[2], a renowned scientific expert on security, defends the view that the software engineering community is doing overall a poor job at securing our global information system:

> To a disturbing extent, however, the kinds of underlying flaws exploited by attackers have not changed very much. [...] One of the most widespread vulnerabilities found recently, the so-called Heartbleed flaw in OpenSSL, was [...] failure to apply adequate bounds-checking to a memory buffer.

and that this is mostly avoidable by putting what we know works to work:

> There has been substantial progress in the past 20 years in the techniques of static and dynamic analysis of software, both at the programming language level and at the level of binary analysis. [...] It would be feasible for a building code to require evidence that software for systems of particular concern (for example, for self-driving cars or SCADA systems) is free of the kinds of vulnerabilities that can be detected automatically in this fashion.

to the point that most vulnerabilities could be completely avoided by design if we cared enough:

> Indeed, through judicious choice of programming languages and frameworks, many kinds of vulnerabilities can be eliminated entirely. Evidence that a specified set of languages and tools had indeed been used to produce the finished product would need to be evaluated.

Shocking! Or so it should appear. But the reality is that we are now used to not being able to rely on software in our everyday lives.

[...]

[1] http://cacm.acm.org/magazines/2015/2/182641-we-need-a-building-code-for-building-code/abstract

[2] http://en.wikipedia.org/wiki/Carl_Landwehr

Tutorial: ARM Cortex-Mx

From: Maciej Sobczak <maciej@msobczak.com>
Date: Wed, 29 Apr 2015 05:38:21 -0700
Subject: Ada on Cortex-M - tutorial progress
Newsgroups: comp.lang.ada

Some time ago I have announced here a tutorial for Ada programming on ARM Cortex-M microcontrollers:

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

I am glad to say that ~15 chapters later, with the recently added:

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

this tutorial is slowly approaching completion and I would like to ask you if you can identify any obvious omissions in its coverage. Of course, the intent of this tutorial was not to be a complete guide (neither for the language nor for any given board), but rather something that can show newcomers how to explore available resources so that they can solve new problems on their own. Still, a peer look can reveal gaps that the author was not aware of making.

You feedback is very welcome.

[See also “Tutorial: Arduino Due (ARM Cortex-Mx)”, AUJ 36-1, p. 19. —sparre]

AVR-Ada: Read and Write NFC Tags

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Thu Apr 30 2015
Subject: Read, write, and emulate NFC tags using Adafruit PN532 breakout board, Olimexino-328, and AVR-Ada
URL: http://arduino.ada-language.com/read-write-and-emulate-nfc-tags-using-adafruit-pn532-breakout-board-olimexino-328-and-avr-ada.html

Recently, I have been playing with PN532 breakout board from Adafruit to read NFC tags and to communicate with NFC-enabled devices, like smart phones.

Adafruit's PN532 breakout board uses 3.3V voltage level and it is little tricky to use it with normal Arduino. So I ended up using Olimexino-328, which allows you to switch between 3.3V and 5V operation.

Setup

Olimexino-328 has special UEXT connector, so I created a small adapter board to connect PN532 breakout board to any board with UEXT, including Olimexino-328.

PN532 supports different communication options like I2C, SPI and UART. I am using SPI, since it is relatively easy to setup and if needed, software SPI is also easy to do.

NFC tags

NFC tags come in various forms and types.

For now, I have written code for NFC Forum type 2 and NFC Forum type 4 tags. It would be also relatively easy to support proprietary NXP Mifare Classic tags, but I haven't had time to add code for them yet.

[...]

Ada Inside

AZip - A Portable Zip Archive Manager

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Sat, 28 Mar 2015 13:22:02 -0700
Subject: Ann: AZip v.1.26
Newsgroups: comp.lang.ada

Following the corresponding improvement in the underlying Zip archive library (Zip-Ada), there is a new version of AZip providing inclusion of items into an archive, encrypted with a password.

AZip is a Zip archive manager designed with the goal of keeping the interface as simple as possible - well at least it is the hope...

It offers a few original features, like an in-archive search function. Instead of unpacking the archive to a drive and then search keywords with the command line or a faulty Explorer search, you search with AZip, within the archive, and the number of hits is displayed in the "Result" column (another original feature) with possibility of sorting - like with other columns.

The AZip web page is here, with a link to downloads, code, news, ...

http://azip.sf.net/

[See also “AZip - A portable Zip Archive Manager”, AUJ 34-2, p. 73. —sparre]

Hypervisor Detection Tool

From: Daniil Baturin <daniil@baturin.org>
Date: Sun Apr 5 2015
Subject: A hypervisor detection tool, this time in Ada 2005

It's often handy to include information about hypervisor in tech support reports, and there are already tools that can detect if a system is virtualised and what hypervisor it's running on, but I don't like them much.

One popular tool is virt-what which is a mix of C and shell that requires root privileges (and only works on Linux), another one is imvirt which is a mix of C and Perl.

Whether "I don't like the language" is a valid motivation for writing a new tool or not, I made one in Ada.

At this point it can detect anything that uses CPUID hypervisor leaf on any OS; Xen PV and HVM on either Linux or FreeBSD; VirtualBox, Parallels, and MS Virtual PC on Linux.

What's missing: proper build setup, container virtualisation detection.

Code review and patches are welcome.

The source: https://github.com/dmbaturin/hvinfo

BIOS Implementation

From: Edward O'Callaghan (funfunctor)
IRC-network: Freenode
IRC-channel: #Ada
Date: Tue Apr 21 18:29:00 CEST 2015

x86 BIOS written in Ada. Look ma', no RAM needed!

[edward@tinypuppy src]$ qemu-system-i386 -bios cpu/aperture.rom -nographic

Booting *** Aperture *** firmware 3ed0e42-UNCLEAN (GNAT GPL 2014 (20140331)).

qemu: terminating on signal 15 from pid 14425

[https://github.com/victoredwardocallaghan/aperture/]

Internship: Multi-core Software Timing

From: Rapita Systems
Date: Thu Apr 23 2015
Subject: Internship: Multi-core software timing
URL: http://www.rapitasystems.com/about/careers/internship-multi-core-software-timing

Rapita Systems is a small and friendly high-tech software company in York (near University) that develops software tools for on-target software verification, optimisation and code coverage of critical real-time embedded systems.

The technology developed by Rapita Systems Ltd targets the aerospace and space industries. Please see http://www.rapitasystems.com/ for more information.

Project details:

This project will investigate the real-time behaviour of multi-core processors, which is an area of growing interest for avionics systems development.

The project will involve writing or selecting some benchmark code for a multi-core system and measuring its

performance under a variety of situations, which will provide Rapita with vital data for demonstrating solutions to customers. We have a number of high-end embedded multi-core embedded systems to evaluate including P4080, AURIX, LEON and we would like to understand the impact of running our software verification tools on these multi-core processors.

This is a project that would suit someone looking to get into low-level and embedded software, real-time systems or software verification technologies. This project has a research slant, with opportunities to steer the work in a variety of directions.

We are looking for:

- Excellent software skills and experience in C (and/or Ada), especially with the idea of "bare metal" programming.
- Familiarity with multi-core concepts will be valuable.
- A hardworking, proactive and diligent student who would relish the opportunity to work on a highly technical project involving programming, design and test.

What you can gain from this internship:

- Experience of working within a fast paced technology company, on industrial research and practical problems.
- The opportunity to help a cutting edge software company reach out to new and existing customers.
- The opportunity to contribute to the area of multi-core usage in aerospace, and publications on this work.
- The opportunity to gain quality, project-based work experience to enhance your CV and employability prospects.
- Opportunity for part time and full time employment.

[...]

Internship: Requirements Management Tool

From: Rapita Systems
Date: Thu Apr 23 2015
Subject: Internship: requirements management tool for certification and qualification of software for aerospace
URL: http://www.rapitasystems.com/about/careers/internship-requirements-management-tool-certification-and-qualification-software

[...]

Project details:

Rapita provides a DO-178C “qualification package” (a set of tests and documentation that we deliver to customers that show that our tools meet their requirements in the aerospace software domain). The project involves finding and adopting a software tool that will manage the requirements and supporting processes for the ongoing development and maintenance of this qualification kit.

The project will start by understanding Rapita’s needs for requirements management and performing an evaluation of some existing options. If a suitable existing tool is found then the project will evaluate and roll out this tool, adapting it to Rapita’s needs as necessary. If no suitable tool is found then the project may design and start to implement a custom tool.

The project will involve requirements management, processes and working with safety-related software and would suit someone who likes formality, processes and correctness.

We are looking for:

- Someone with excellent software skills, ideally including C, Perl, Ada, bash/scripting.
- Experience of Windows/Linux administration, and ideally Javascript/AJAX.
- An understanding of software process, requirements and safety-critical software
- A hardworking, diligent and proactive student who enjoys working as part of a team.

What you will gain from this internship:

- Experience of working within a small, friendly, fast paced technology company, on industrial research and practical problems.
- The opportunity to help a cutting edge software company reach out to new and existing customers.
- Experience of working with industrial experts and with leading aircraft manufacturers in a challenging and exciting domain.
- The opportunity to gain quality, project-based work experience to enhance your CV and employability prospects.
- Opportunity for part time and full time employment.

[...]

Internship: Software Quality for Safety-Critical Systems

From: Rapita Systems
Date: Thu Apr 23 2015
Subject: Internship: Software Quality for Safety-Critical Systems
URL: http://www.rapitasystems.com/about/careers/internship-software-quality-safety-critical-systems

- Opportunity for part time and full time employment.

Rapita Systems is a small and friendly high-tech software company in York (near University) that develops software tools for on-target software verification, optimisation and code coverage of critical real-time embedded systems. The technology developed by Rapita Systems Ltd targets the aerospace and space industries. Please see http://www.rapitasystems.com/ for more information.

Project details:

The project involves the verification of safety-critical software tools and is an excellent introduction into practical software engineering for reliable software.

The main aim of this project is to support the development and maintenance of the DO-178C qualification kit for our software (a set of test cases and documentation that are delivered to our customers that shows that our software meets its requirements and is able to be used for aerospace/avionics software).

Part of the work will involve the creation of formal tests in C and Ada, leading to the diagnosis and fixing of bugs, working with the test and development teams to improve the software, and performing integrations of the software with various embedded platforms such as P4080, PowerPC and other platforms used in aerospace software.

This project is quite a general project involving a variety of skills can be tailored to the skills of the student. This project would suit someone keen on formal correctness and looking to develop experience in reliable software.

We are looking for:

- Excellent software skills, ideally using C, Perl, Ada, bash/scripting
- Linux/Windows administration
- An interest and understanding of software process, requirements and safety-critical software
- The ability to work in a team with a hardworking, proactive and diligent attitude

What you will gain from this internship:

- Experience of working within a fast paced technology company, on industrial research and practical problems.
- The opportunity to help a cutting edge software company reach out to new and existing customers.
- The opportunity to work with industrial experts and leading manufacturers in a challenging and exciting domain.
- The opportunity to gain quality, project-based work experience to enhance your

[...]

Job: Writing a Binding to Open Z Wave

From: Tony G. <tonythegair@gmail.com>
Date: Sat, 2 May 2015 04:17:16 -0700
Subject: open source ada binding to zwave
Newsgroups: comp.lang.ada

I have some funding for an energy/carbon saving project, I am completing in Ada and Gnoga. I have a requirement for an Ada binding to an existing Open Z Wave library, and am looking for someone who has the skills and ability to do that. I can pay! Not a lot! But I can pay! And the result would be an Open Source library for Ada that would be a Debian package for ARM and Intel.

This package would allow me to eliminate a dependency and I believe improve the reliability of what I am doing as well as being able to complete the project entirely in Ada...Hooray! It is highly likely that the result could be a learning resource for incoming programmers!

Any advice for the protection of worker and commissioner towards completing this piece of the project I would be grateful for!

http://www.openzwave.com/

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sat, 02 May 2015 10:30:17 -0700
Subject: Re: open source ada binding to zwave
Newsgroups: comp.lang.ada

> [...]

It does look painful. How much are you willing to pay?

From: Tony G. <tonythegair@gmail.com>
Date: Sat, 2 May 2015 12:28:47 -0700
Subject: Re: open source ada binding to zwave
Newsgroups: comp.lang.ada

If you have to ask, I probably cannot afford it :) but I can stretch to possibly 3k dollar (about 2k in stirling)

Seriously though, first I am trying to ascertain the size of the job etc. I will make the library available to other users and it may be the case that if someone made a decent start on it, I could finish the major part of it myself. It would be to support a social enterprise trying to reduce peoples energy bills, but I do appreciate that people need to pay bills and eat etc.

One way might be to use the gnatcoll library and then maybe use python scripts through this library but I think it is preferable to have an Ada binding.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 2 May 2015 21:42:49 +0200
Subject: Re: open source ada binding to zwave
Newsgroups: comp.lang.ada

> [...]

Where is a problem? They seem have drivers, use these, if you really need to support these devices. It does not look like a big deal.

Ada in Context

Tagged Type Abuse

From: Natasha Porté <lithiumcat@instinctive.eu>
Date: Thu, 18 Dec 2014 11:26:58 +0000
Subject: Tagged type abuse
Newsgroups: comp.lang.ada

I find myself using more and more tagged types for reasons that have nothing to do with tagging, mostly the prefix notations (when it helps readability and when a function call is conceptually accessing a record element but with a hidden concrete implementation) and the passing by reference.

However, I still feel guilty about it, like I'm abusing a feature unrelated to what I wish to accomplish.

What would you recommend to appease such feelings?

Sacrificing prefix notation readability on types that have no business being tagged?

Trying to be more pragmatic and use tools (and language features) for any purpose at which they end up being useful, even unintended?

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Thu, 18 Dec 2014 09:59:36 -0700
Subject: Re: Tagged type abuse
Newsgroups: comp.lang.ada

> [...]

There are three reasons I use tagged types:

1. To obtain finalisation

2. To avoid explicit pointers in self-referential types

3. To obtain Object.Operation notation

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 18 Dec 2014 17:10:37 -0600
Subject: Re: Tagged type abuse
Newsgroups: comp.lang.ada

> [...]

I think that Jeff is saying that not everyone thinks dynamic dispatching is relevant. But tagged types still can be useful.

Note that in pre-2012 Ada, you needed to use tagged types get "=" to work right vis-a-vis composition. (Ada 2012 extended that to all record types, which of course means that some programs that expect the wrong answer will break. But in most cases, the change will fix bugs rather than create them.)

I don't use dynamic dispatching much (outside of Claw anyway), but I sometimes use inheritance to inherit implementations (rather than having to duplicate them all over, with the corresponding maintenance headache).

I'd probably use tagged types to get prefix notation, but I'd have to implement it in Janus/Ada first. :-)

Anyway, I wouldn't worry about it. Tagged types cost about the same as regular record types (the only difference is the waste of space for the tag, which only matters for tiny records) unless you use T'Class. So do what makes you program work better.

(We couldn't make cursors in the containers be tagged, thus you can't use prefixed notation to do various reading operations on the containers. One more reason out of many that I think every container operation should have had a container parameter. [Another reason is that preconditions make much more sense if the container passed to an operation has a name.] But of course there is as many container designs as there are programmers -- perhaps more -- and the big value was picking one. There is no way it could have been perfect for every use anyway.)

{Pre,Post}conditions and Side Effects

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 22 Dec 2014 17:46:28 -0600
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

[...] It is one reason that a package author can prevent some or all assertions from being disabled in the package.

> [...]

A much better example is an implementation that has implemented the containers using preconditions rather than explicit checks. For instance:

```
procedure Replace_Element
          (Container: in out Vector;
           Position: in Cursor;
           New_Item: in Element_Type);
   with Pre'Class
        => (if Tampering_With_Elements
_Prohibited (Container) then
      raise Program_Error) and then
      (if Position = No_Element then
      raise Constraint_Error) and then
      (if not Cursor_Belongs_To_Container
            (Container, Position) then
     raise Program_Error);
```

If this is called with preconditions ignored, the required semantics of Replace_Element won't happen (because the checks for the various exceptions won't happen - no one is going to repeat the precondition checks in the body - if

that was required, the precondition is worthless). (Note: This precondition uses a couple of predicates that aren't defined in the Ada 2012 containers, but should have been. Most likely, the next version of Ada will rewrite the containers this way, it will get rid of a lot of text in the Standard.)

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Tue, 23 Dec 2014 18:02:42 -0500
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

[...]

A simple example that I love because it is so simple yet so telling is binary search of an array. A reasonable precondition is that the array it is given is sorted. It might look like

```
Pre => (for all I in A'First .. A'Last - 1 =>
          (A(I) <= A(I + 1)))
```

This takes O(n) time to evaluate. Yet binary search is an O(log(n)) algorithm. For large arrays the precondition might take many thousands or even millions of times longer to execute than the subprogram itself.

[...] Thus putting anything resembling essential program logic in an assertion is, of course, just wrong. Forbidding assertions with side effects might be nice, but the programmer still has to be careful with them anyway.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Tue, 23 Dec 2014 20:03:37 -0500
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...] A reasonable precondition is that the array it is given is sorted. [...]

Yes, all that's true. But those would be better as predicates/invariants instead of preconditions. For example, if you know Sort produces a sorted array, and Binary_Search takes a sorted array, you don't have to check for sorted-ness on entry to Binary_Search if you've got a Sorted_Array.

But note that your "Pre" above is a good example of what I was saying in a somewhat-unrelated post in this thread: It is shorter and simpler than doing a sort using some efficient sorting algorithm.

> For this reason I assume that in most cases programs must be deployed with assertions disabled or else there is little chance the program will be able to meet its performance goals.

Yes, or at least SOME assertions disabled.

I like to say, "If you don't need to disable assertions, then you don't have enough assertions". It's a silly sound bite that is not always true, but there's a grain of truth in it.

[...]

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Fri, 24 Apr 2015 10:59:52 +0200
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...] Thus putting anything resembling essential program logic in an assertion is, of course, just wrong.

But when are you putting "essential program logic" in an assertion?

```
1) subtype Non_Negative_Matrix is
Ada.Numerics.Real_Arrays.Real_Matrix
   with Dynamic_Predicate  =>
     (Non_Negative_Matrix'First (1) = 1) and
     (Non_Negative_Matrix'First (2) = 1) and
      for all E of Non_Negative_Matrix =>
          E >= 0.0);

2) procedure New_Line (File : in File_Type)
   with Pre => Is_Open (File) and then
     Mode (File) in (Out_File | Append_File);

3) function Binary_Search (Source : in List;
                           Key    : in Keys) return Values
   with Pre => Sort (Source);
   --  Sorts Source if it isn't already sorted.
```

I consider examples (1) and (2) fine, but example (3) a very bad idea.

At the same time, I know that my application may fail silently if the assertion in example (1) isn't true.

When it comes to example (2), I expect that the operating system (if nothing else) will make sure that my application doesn't fail silently if the assertion isn't true.

But I dislike banning "essential program logic" in assertions, as any assertion is program logic. And if it isn't essential, why should it be there?

One problem I have with assertion aspects is that I get the same exception no matter which mistake I have made. If I put the check inside a subprogram instead of in its profile, I can get more clear information about which kinds of mistakes I make.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 24 Apr 2015 11:18:28 +0200
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...] I get the same exception no matter which mistake I have made. [...]

Your wishes will be soon satisfied, see AI12-0022-1 and AI12-0054-2 (raise expression and aspect Predicate_Failure)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu Jul 10 2014
Subject: Version 1.13 of ai12s/ai12-0022-1.txt
URL: http://www.ada-auth.org/cgi-bin/cvsweb.cgi/ai12s/ai12-0022-1.txt?rev=1.13

[...]

!wording

Add to 4.4(3/3):

| raise_expression

Rename 11.3 to "Raise Statements and Raise Expressions"

Add before 11.2(6):

An exception_name of an exception_choice shall denote an exception.

Add after 11.3(2/2) [Syntax]

raise_expression ::= raise *exception*_name [with *string*_expression]

[...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 24 Apr 2015 18:39:24 -0500
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...]

Raise expressions have been implemented in GNAT for quite a while; they probably exist in the compiler you're using. Predicate_Failure wasn't implemented until very recently (after someone wrote an ACATS test for it), so for that you might have to wait.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Fri, 24 Apr 2015 14:10:19 +0200
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> But I dislike banning "essential program logic" in assertions, as any assertion is program logic. And if it isn't essential, why should it be there?

This definition involving "essential" does not reflect "contract" properly, IMHO. And it is fallacious in that it fails to reflect the particulars that make assertions, as in "assertion as per contract", different from just program logic. To see this, I think it is helpful to free oneself of the limitations of looking at assertions from just a programmer's point of view.

Assertions (of the contract) should never, ever be understood to be consequences of the program text with or without the assertions, insofar as they are agreed upon by humans to be true about the program to be, its intent in particular. They cover a program that may even have to be written yet. They do share some of the properties of agreements manifest in specifications.

For contracts' clauses, possibly supported by assertions as part of source text, there isn't even a conceptual necessity to have proper assertions tested in the very same environment as the program proper: a copy will do in many cases of proper assertions, as these are pure Boolean functions.

The fact that assertions can be expressed in Ada is purely accidental; SPARK shows that a different language can be used, and may even be more expressive. Comments do count, too, such as RM statements about O(op). The latter can be understood as a part of contract between any user of Ada and the implementer of Ada.

An improper assertion (if I may call it that for reasons of delineation) will modify that part of program's data which is covered by the contract, data to be handled solely by the program which would yield the same effects that are stated in the contract, with or without assertion checking. So, using improper assertions, you'd be making a mess, even though results might come out right (only deferring proof obligations having to do with the improper assertion).

OTOH, whenever testing an assertion requires computation, it is essential to keep its doings separate from what the program needs to compute to fulfill the contract.

So, the idea of considering assertions of contracts (as opposed to plain old debugging asserts) from the viewpoint of their implementation is misleading.

Illustration:

Company X agrees this is in a contract:

If, before calling Binary_Search, input is sorted, then the result of calling Binary_Search will be ...

That's a statement that can be made part of a contract, and it may be formally reflected in Ada aspects, if possible. It is expressing the idea that Source is sorted.

Last but not least, a precondition should never be anything but an assumption. As not checking it at run-time is a valid way of handling preconditions, the outcome of not testing should still not create havoc; so, the caller needs to make sure that assertion of the contract is always true, and never depend on how it is tested, or on that it is tested.

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Fri, 24 Apr 2015 16:40:24 +0200
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...] the caller needs to make sure that assertion of the contract is always true, and never depend on how it is tested, or on that it is tested.

Taking that view, there isn't any point combining the contract notation of Ada 2012 and SPARK 2014, as it would prevent you from writing a single source text which was valid for both languages.

SPARK would complain about your in-subprogram check (mirroring the precondition), as raising an exception is illegal in SPARK (and dead code probably is as well).

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Fri, 24 Apr 2015 18:29:51 +0200
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [dual-language source text]

That's hardly possible anyway, given the number of restrictions that SPARK 2014 imposes, even when not taking assertions into account. Is

--# hide

all gone?

Also, why would there be a technical need to have a contract use only Ada or SPARK 2014 as the single language? That's not done in the LRM, which uses logic and mathematics; writing SPARK 2014, non-SPARK compilers could simply omit analysis of the language used in Pre => ... etc., presuming they can handle the syntax (likely, I should think). And some things cannot reasonably be stated formally anyway. (I guess it becomes apparent that maybe a combined language turns into a combined stricture! ;-)

The makers of, respectively, GNAT and SPARK have merged, that seems like a start for better merging the languages; Tucker Taft, now also at AdaCore, has alluded to excessive restrictions in SPARK. The definitions of SPARK had added more of Ada over the years already (tasks, tagged types, ...).

So, I guess, in the long run, there is no more risk of Ada programs that suffer from conflicting language desires than there is now when source texts show non-Ada 'Img and GNAT is not your Ada compiler. ;-)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 24 Apr 2015 18:46:27 -0500
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...]

Besides, Ada 2012 has a mechanism to ensure that preconditions are in fact evaluated; that exists for this very reason. If you have a local Assertion_Policy of Check, that applies to the declarations and the precondition ought to be evaluated no matter what policy is in effect at the point of the call. (Whether GNAT actually gets this right is unknown.)

Otherwise, one could not "hoist" the various rules for I/O, containers, and the like into preconditions. Which would seem like madness. Certainly checking the same thing twice (which is what would happen if you put the condition into the precondition and then manually checked it a second time in the body) is madness.

There is a camp that thinks that ignoring contract assertions is very similar to suppressing checks, and anything that happens after doing that is effectively erroneous. (That's NOT the wording in the Standard.) For that group, hoisting things into preconditions is fine. Otherwise, one needs to take steps to ensure that they're evaluated.

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Fri, 24 Apr 2015 18:26:41 -0400
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> But when are you putting "essential program logic" in an assertion?

I think a servicable rule is this: If the program works as required, with all necessary checks still present, with all assertions removed, then we can say the assertions contain no essential program logic.

In a correct program all assertions should always be true.

> 1) subtype Non_Negative_Matrix is Ada.Numerics.Real_Arrays. Real_Matrix

> with Dynamic_Predicate

> => (Non_Negative_Matrix'First (1) = 1) and

> (Non_Negative_Matrix'First (2) = 1) and

> (for all E of Non_Negative_Matrix => E >= 0.0);

Although the Dynamic_Predicate asserts that the matrix elements are all non-negative, this does not remove the program's obligation to include checks that no negative elements are added to the matrix. The assertion only exists to catch mistakes in those checks. It does not exist to actually *be* those checks. In that respect the assertion is not "essential program logic."

> 2) procedure New_Line (File : in File_Type)

> with Pre => Is_Open (File) and then

> Mode (File) in (Out_File | Append_File);

Similarly here the program is still obligated to only pass File objects to New_Line that represent open files. If the program accidentally passes an unopened file to New_Line the assertion will catch the logical error. However, the assertion should not take the place of earlier checks. Again the assertion is not essential program logic.

[...]

I agree that (1) and (2) are fine, but that doesn't mean the program should rely on the assertions for its proper functioning.

The assertions check correctness; they don't implement it. Even if the assertions are removed, the program should still execute properly.

> [...] And if it isn't essential, why should it be there?

Because we often make mistakes and it's nice to have our thinking double checked. Also, of course, the assertions make our intentions known to tools, such as SPARK, that can automatically verify our code implements the conditions we are asserting.

> [...]

Putting the check inside the subprogram is quite a different thing. That is part of your implementation of correctness. Since assertions should never fail, using the same exception for all of them isn't terrible. That said, the upcoming feature that allows different exceptions to be used when an assertion fails is nice too.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 24 Apr 2015 19:13:14 -0500
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> In a correct program all assertions should always be true.

Sure, but that applies to lots of other things, too. For instance, in a correct program, Constraint_Error or Program_Error should not be raised. But it still happens.

> [Comments to example 1 and 2.]

I definitely disagree here. This example (2) is essentially similar to the one given in the upcoming Corrigendum (3.2.4(41-51/4). In a case like this, the precondition (or predicates as in the example) *replace* the checks required by English text in the RM. There would no internal checks of correctness.

You are of course correct that no caller should call New_Line with a closed file, but that's irrelevant because it can happen anyway (there is no static way to prevent it). There has to be code somewhere to handle it. So, in such a case, a precondition serves two purposes: (1) to signal to the client what conditions are expected, and (2) to determine what happens if those conditions aren't met. (2) certainly is "essential program logic", at so far as one cannot meet the published specification of New_Line without it.

Ada prior to Ada 2012 has a problem in that the reasons an exception can be raised conflate the programmer mistakes with conditions that are impossible for the programmer to know (consider the difference between whether a file object is open vs. whether a file exists on the disk). Preconditions and predicates provide a way to separately specify the first kind of situation vs. the second kind. (Ultimately, one hopes, compilers will be able to eliminate much of the runtime checking associated with preconditions and predicates, which is not possible in the pre-Ada 2012 world.)

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Fri, 24 Apr 2015 21:01:00 -0400
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...] In a case like this, the precondition (or predicates as in the example) *replace* the checks required by English text in the RM. [...]

In this case it's not the internal checks I mean. Hoisting the internal checks into preconditions makes sense to me, at least in certain (many?) cases. In my comments above I'm talking about checks occurring before New_Line is called.

Somewhere the programmer tried to open a file. If the programmer attempts to call New_Line without first verifying that the file opened successfully, that's a logical error in the program. Checking that Is_Open (File) is true provides some protection against such an error... regardless of if the check is a precondition or done inside the body of New_Line. Either way, in a correct program that check should never fail. The beauty of doing it in a precondition is that the "unnecessary" check can be removed by changing the assertion policy.

In contrast imagine a procedure that takes file and does some processing on it. Suppose the procedure raises some exception if the file has the wrong format. The programmer might decide that it's not wrong to call the procedure with an incorrectly formatted file, and let that be a matter for the procedure to worry about. In that case, adding the check as a precondition doesn't seem right; a correct program might call the procedure with a badly formatted file.

On the other hand if the programmer decides it's illogical to call the procedure with an incorrectly formatted file because the file has (supposedly) been verified previously, using a precondition to check the format makes sense.

Same procedure, same check... the sensibility of making the check a precondition depends on the context in which the procedure is used. In the first case the caller relies on the procedure to do the check. In the second case the procedure relies on the caller to do it. Ultimately it ends up being a design decision.

From: Bob Duff <bobduff@theworld.com>
Date: Fri, 24 Apr 2015 20:31:44 -0400
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...]

I think what Peter meant by "essential program logic" is code that, if deleted from the program, would cause the program to malfunction.

> [...]

The assertions in (1) and (2) are not "essential program logic"; if you delete them, the program will still work properly. That's fine -- you should write assertions so that deleting them from a correct program will have no effect.

[...]

> [...] But I dislike banning "essential program logic" in assertions, as any assertion is program logic. And if it isn't essential, why should it be there?

Same reason we put comments in the code. Comments are not "essential program logic" in the sense defined above -- if you delete all the comments, the program will still work. But we still want comments. Likewise, one should normally write assertions (like Pre and Predicate) so the program still works if they are deleted.

Assertions are like comments, except we have a higher confidence that they are actually true.

> [...]

You can say:

```
Pre => X = Y or else raise
          X_Not_Equal_To_Y;
```

From: Vincent Diemunsch <vincent.diemunsch@gmail.com>
Date: Sat, 25 Apr 2015 05:08:32 -0700
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> [...] Assertions are like comments, except we have a higher confidence that they are actually true.

I agree. Assertions express logical properties of the program. One can have a high confidence in them for two reasons:

1. mathematical correctness according to a given theory

2. proof that the code is coherent with the assertion, using a tool.

But Assertions should stay as comments, for they are not code but logical formula expressed in a mathematical language. They were comments in SPARK 2005 and it is still the case in Frama-C or many formal proof systems. Hoare logic is supposed to give "correctness by construction": Which is the ability to never fail on a runtime test. This is required in safety critical systems.

But what Ada and SPARK 2014 are doing is "design by contract", as Bertrand Meyer called it. This makes a confusion between a precondition and a runtime test. It may look appealing in the beginning but it is nothing else than a test harness put around a subprogram. With all the problems

related to it: How to debug it? Should it raise exceptions? It breaks the separation between specification and implementation etc.

Therefore, I don't think that it is the right choice for a language that is mainly used in safety critical systems.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Sat, 25 Apr 2015 18:37:13 +0200
Subject: Re: {Pre,Post}conditions and side effects
Newsgroups: comp.lang.ada

> But what Ada and SPARK 2014 are doing is "design by contract", as Bertrand Meyer called it. This makes a confusion between a precondition and a runtime test.

Actually, Meyer insists that contracts have an associated notion, that of proof obligation. And the description of DbC is declaring quite openly that not needing to perform run-time tests (e.g. defensive programming) is a design goal.

What you get in the lesser (than some fancy ideal) situation is summarized in a table of OOSC2, §11.6 (coincidence? ;-), for a stack:

Put OBLIGATIONS BENEFITS

Client satisfy Pre=>... from Post=>...

Only call Put(X) on Get stack updated: not a non-full stack. Empty, X on top, Item yields X, Count increased by 1.

Supplier satisfy Post=>... from Pre=>...

Update stack reprsntn Simpler processing thanks to have X on top (Item to the assumption that yields X), Count
stack is not full.
increased by 1, not
Empty.

Looking at SPARK 2014, it seems to not have changed earlier SPARK WRT being a tool for analysis before run-time.

Ada, OTOH, looks like becoming a programming language facilitating either type of checking, as before, but more extensively and more formally, and more of it to write for the programmer.

Debug “Macros”

From: Brad Moore <brad.moore@shaw.ca>
Date: Tue, 30 Dec 2014 13:07:41 -0700
Subject: Re: Any Suggestion How To Accomplish A Debug Macro?
Newsgroups: comp.lang.ada

> [...] isn't there a way to make some sort of debug macro? [...] without surrounding it with an if statement and a Boolean flag?

[...] declare a static constant somewhere, and use the value of that constant to decide if logging should occur. If the compiler/linker supports dead code elimination, then the debug code can be eliminated if that variable is set to False.

Eg.

```
package Debug_Logging is
   Debug_Enabled : constant Boolean :=
         False;  -- Edit this line
   procedure Log (Message : String);
end Debug_Logging;

with Debug_Logging;
procedure Foo is
begin
   -- if statement removed if
   -- Debug_Enabled is false
   if Debug_Enabled then
     Log ("Entered Foo");
   end if;
end Foo;
```

This works in GNAT, and might work in other compilers as well. Worst case is that the Debug_Enabled Boolean get evaluated in multiple places, but that overhead of evaluating a Boolean might still be acceptable for a compiler that doesn't do dead code elimination.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 30 Dec 2014 16:11:56 -0600
Subject: Re: Any Suggestion How To Accomplish A Debug Macro?
Newsgroups: comp.lang.ada

> [...]

That's of course the Ada way. What's the point of avoiding the Ada way here? Everything in Ada is more verbose than C -- some of us think that's the advantage of Ada (more for readability than writability).

[...]

I usually make these flags more complex (like an array of constants) so that various sets of tracing can be enabled in order to track whatever is wrong.

Indeed, in most of my programs, I use a flag set at runtime (controlled either through the GUI or through command-line switches. In that case, the code is always there (but a Boolean test is cheap), but that means I don't have to waste time with a compile-link-test-repeat cycle to trace a problem. (I admit, it's often necessary to add additional tracing to actually find the root cause, but the initial tracing at least can narrow it down quickly.)

> [...]

I think that the set of compilers that don't do dead code elimination is close to the empty set. (That was pretty much the first optimisation we did in Janus/Ada, even before we had packages or floating point.) Whether the dead code elimination can get rid of everything (it won't get rid of string literals in Janus/Ada, for instance) is a different question.

But I don't think there is much reason (outside of the memory-constrained embedded system, or the system that has to be formally proved or validated) to ever removing the tracing. It's important to be able to turn it off, of course, but the runtime cost of it being off is so minimal (primarily caching/paging effects) that removing it isn't worth the effort. (And if you plan to keep it around forever, you'll spend more time making the traces make sense in the future -- which typically pays off very quickly.)

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 02 Jan 2015 21:37:47 +0100
Subject: Re: Any Suggestion How To Accomplish A Debug Macro?
Newsgroups: comp.lang.ada

> But I don't think there is much reason [...] to ever removing the tracing. [...]

I fully agree with that.

In AdaControl, there is a sophisticated tracing capability, and it's enabled with a command line option. If a user has a problem, he just has to rerun the program with -x and send me the output - that's usually enough to identify the problem. I can't tell how much time it saved me. And since AdaControl spends about 65% of its time in ASIS, the cost of testing a Boolean is negligible.

Expression Functions in Protected Types

From: Simon Wright <simon@pushface.org>
Date: Tue, 20 Jan 2015 22:34:53 +0000
Subject: Use of expression function in protected type
Newsgroups: comp.lang.ada

Is it permissible to use an expression function as the completion of a protected function? (GNAT thinks so).

```
  protected Button is
     function Current_Index
           return Interval_Index;
  private
     Index : Interval_Index := 0;
     procedure Handler;
     pragma Attach_Handler (Handler,
          Ada.Interrupts.Names.EXTI0_IRQ);
  end Button;
and then
  protected body Button is
     function Current_Index return
Interval_Index is (Index); -- <<<<<<<<
     procedure Handler is
     begin
        HAL_GPIO_EXTI_IRQHandler (
           16#0001#);
        Index := Index + 1;
     end Handler;
  end Button;
```

ARM 6.1(30) distinguishes an expression_function_declaration from a subprogram_declaration.

6.8(4) allows an expression_function_declaration to be a completion.

9.4(8) says a protected_operation_item can be, inter alia, a subprogram_declaration or a subprogram body. (Subprogram_declaration? How can that be?)

From: Egil Harald Høvik <ehh.public@gmail.com>
Date: Wed, 21 Jan 2015 00:35:51 -0800
Subject: Re: Use of expression function in protected type
Newsgroups: comp.lang.ada

> [...]

Just like a package can have subprograms declared in the public part, private part or the body, protected subprograms can be declared in the public part, private part or body of a protected type. (For example, it's not uncommon for a barrier function to be declared in the body.)

And just like in a package body, forward declarations (or subprogram_declarations) are allowed for subprograms.

However, an expression_function is allowed to complete a subprogram_declaration, but is not itself a subprogram_declaration (ARM 6.1(30/3). As far as I can tell, it's a basic_declaration, which is allowed in package specifications and bodies, but not in protected_operation_items.

From: Simon Wright <simon@pushface.org>
Date: Wed, 21 Jan 2015 09:15:01 +0000
Subject: Re: Use of expression function in protected type
Newsgroups: comp.lang.ada

> [...]

One lives and learns! I've never had occasion to write a barrier _function_; my most complex barrier was:

```
when A
  or else not B
  or else (C and then not (D and then E))
```

> [...] but not in protected_operation_items.

That was my reading, but I wondered whether it was deliberate, an oversight, or I'd missed something.

The context was Emacs ada-mode, whose indentation engine is built on a parser, which follows the ARM.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 21 Jan 2015 14:44:40 -0600
Subject: Re: Use of expression function in protected type
Newsgroups: comp.lang.ada

> [...]

> That was my reading, but I wondered whether it was deliberate, an oversight, or I'd missed something.

Certainly not deliberate, I'm pretty sure no one ever considered it. We will now (I've forwarded a version of your message to Ada-Comment).

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 21 Jan 2015 20:50:40 -0600
Subject: Re: Use of expression function in protected type
Newsgroups: comp.lang.ada

> [...]

Early returns are that it is an oversight. It'll be on the agenda for next week's ARG phone call, and quite possibly it will get included in the upcoming Corrigendum. If so, it probably will be close to the fastest official Ada fix ever...

Preventing Errors

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 23 Jan 2015 15:26:55 -0600
Subject: Re: Strange error
Newsgroups: comp.lang.ada

> [...] How do you professionals prevent such stupid errors? [...]

You don't (or at least, I don't). I seem to write loops that don't loop (forgot the P := P.Next) all the time.

Probably the only real difference is that we're used to questioning everything: If faced with a Reverse_Print routine not working, we'd be quicker to consider that the input might not be correct. (Indeed, I'd probably start with that assumption, because the display routine is so simple.) But there is no certainty that we'll look in the right place.

That's, of course, one of the reasons we're interested in Ada, because it's possible to move more mistakes to compile-time checks. Bugs detected by a compile-time check never need to be debugged from results that might be hard to reproduce. (And as well, Ada lets us more easily put in runtime checks, which prevent problems from lingering.)

From: Bob Duff <bobduff@theworld.com>
Date: Fri, 23 Jan 2015 20:34:04 -0500
Subject: Re: Strange error
Newsgroups: comp.lang.ada

> [...]

"don't loop"? That loops too much. ;-)

I tend to write the boilerplate first:

```
while P /= null loop
   P := P.Next;
end loop;
```

Then go back and fill in the body of the loop. So I don't usually make that particular mistake. Anyway, I think GNAT will give a warning about that.

But in Ada 2012, we have iterators, which largely solves the problem. Put all your eggs in one basket, and if the iterator works, then all the myriad "for" loops around the code will work.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 26 Jan 2015 15:44:33 -0600
Subject: Re: Strange error
Newsgroups: comp.lang.ada

> [...] boilerplate first [...]

I sometimes do that, but sometimes I'm so focused on the important stuff (the body of the loop) that I forget the structure.

> Anyway, I think GNAT will give a warning about that.

It certainly gives warnings on loops that aren't a problem. :-) I've never seen one on a loop that is a problem, but then again, most of my code was written using another compiler first, so most of the gross errors have already been removed.

> But in Ada 2012, we have iterators, [...]

Yeah, but that would mean finding time to implement them in my favourite compiler. :-)

From: Bob Duff <bobduff@theworld.com>
Date: Fri, 23 Jan 2015 19:47:16 -0500
Subject: Re: Strange error
Newsgroups: comp.lang.ada

> [...] How do you professionals prevent such stupid errors? [...]

One way is to use Ada.Containers.Doubly_Linked_Lists. But that won't work for you, because you're not trying to use doubly-linked lists, you're trying to learn how to implement them. Which is something programmers should know how to do.

So draw a doubly-linked list on paper, with circles and arrows. Go through each procedure and "execute" it by hand, erasing the arrows and drawing new ones. Take care to execute what you wrote, not what you meant to write. Bugs like the one mentioned will usually become obvious.

Bounded Vectors, Reference Types, and the Secondary Stack

From: Simon Wright <simon@pushface.org>
Date: Sun, 15 Feb 2015 16:21:43 +0000
Subject: [Bounded] Vectors, reference types, and the secondary stack
Newsgroups: comp.lang.ada

It turns out that (GCC 4.9.1) that if you have

```
package Interval_Containers
   is new Ada.Containers.Bounded_Vectors
     (Index_Type   => Natural,
      Element_Type =>
            Ada.Real_Time.Time_Span,
      "="   => Ada.Real_Time."=");
  Intervals : Interval_Containers.Vector (5);

and then
  Intervals.Insert_Space (0, 5);
  Intervals (0) := Ada.Real_Time.Milliseconds
(50);

then
```

```
function Reference
  (Container : aliased in out Vector;
   Index     : Index_Type) return
           Reference_Type;
```

returns its result on the secondary stack!

Why would it need to do that? given the (private) definition

```
type Reference_Type
        (Element : not null access
        Element_Type) is null record;
```

You ask why I would care. Well, in my STM32F4 RTS the environment task, in which elaboration happens, isn't actually a task, and doesn't (yet) have a secondary stack.

The reason it's not a task is that the way to kick off the FreeRTOS scheduler is to call FreeRTOS.Tasks.Start_Scheduler (aka vTaskStartScheduler()), which doesn't return unless the scheduler can't be started; and I haven't found a way to get this behaviour into the start-up code generated by gnatbind, so the poor user has to call it at the end of their main program.

Story of a GNAT Bug

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Wed, 25 Mar 2015 22:02:24 +0200
Subject: Story of one GNAT bug
Newsgroups: comp.lang.ada

In the past AdaCore has not been that active fixing GNAT bugs so that they are fixed in FSF GCC also. But recently I had totally opposite experience, so I wanted to share the story of my bug.

January 15, 2015, I noticed that ICCAda was rejecting YAMI4-GPL-1.10.0 code with following error:

"yami-parameters.ads", line 808: Error: Private extension has LIMITED keyword, but full type does not. [RM 7.3(10.1)]

After little discussion with Irvine/ICCAda support, I ended up making following test case:

```
-- my_limited.ads
package My_Limited is
   type My_Limited_Type is tagged limited
        private;
private

   type My_Limited_Type is tagged limited
   record
        X : Integer;
   end record;
end My_limited;

-- my_limited_2.ads
with My_Limited;

package My_Limited_2 is
   type My_Limited_Type_2 is limited new
        My_Limited.My_Limited_Type
         with private;
private
   type My_Limited_Type_2 is new
        My_Limited.My_Limited_Type with
     record -- this line should have error
       Y : Integer;
     end record;
end My_Limited_2;
-- END of testcase
```

All tested GNAT versions, including GNAT GPL 2014 accepted the code, so clearly GNAT did not have check for RM 7.3(10.1).

A GNAT Pro owner from #Ada IRC channel confirmed that the bug was present also in GNAT Pro.

Next day (Jan 16), I reported the bug to AdaCore via http://libre.adacore.com/contact/ and it got ID "O116-026 public".

On Feb 5, the fix was pushed to GCC repositories with following changelog entry:

> 2015-02-05 Ed Schonberg <schonberg@adacore.com>

> * sem_ch3.adb (Process_Full_View): Verify that the full view

> of a type extension must carry an explicit limited keyword if

> the partial view does (RM 7.3 (10.1)).

The commit itself is visible at:

https://github.com/gcc-mirror/gcc/commit/31831d39bf4840761c92c9fad5abf29b4feb7b50

So, it took about 3 weeks from the report to have the fix in FSF GCC also.

A week later (Feb 12), I talked to ACAA technical agent about the bug and possibility to add B test for the bug to ACATS. Irvine support people were kept in the loop and they found out some extra time to do the actual test and send it to the technical agent.

The test was accepted and is visible at

http://www.ada-auth.org/cgi-bin/cvsweb.cgi/acats/new/b730010.a?rev=1.1

On March 19, the ACAA technical agent announced ACATS modification list 4.0E and one of the modifications was:

> New test B730010 checks that 7.3(10.1/3) is enforced.

As a result, from now on this bug should be impossible to happen in any Ada compiler.

One should also note how important it is to have multiple Ada compiler implementations (and to have possibility to use multiple of them for the same source code). Without ICCAda checking this, the bug could have been hiding in GNAT for a long time.

PS. I didn't report this to YAMI4 author. The source code had also some other issues and I ran out of free time for a proper bug report.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 25 Mar 2015 16:07:11 -0500
Subject: Re: Story of one GNAT bug
Newsgroups: comp.lang.ada

> [...]

You left out one step here. It turns that not only did GNAT not check the rule in question, but also that there was an ACATS test which expected it to be legal. (That probably happened in part because the test was checking a different rule in 7.3, and using GNAT to check whether it was correct did not turn up the violation of 7.3(10.1/3).)

It's also strange that there wasn't a test for that rule; I thought I had checked all of the new (since Ada 95) rules in 7.3 and that obviously wasn't true. But now it is.

Dynamic Memory Management

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 9 Apr 2015 18:49:40 -0500
Subject: Re: BDD package in Ada.
Newsgroups: comp.lang.ada

[...] Ada provides at least 5 ways to manage dynamic memory:

(1) Stack

(2) Container

(3) Controlled types (as in Smart Pointers)

(4) Subpools (perhaps "semi-manual")

(5) Traditional allocate/deallocate

There's nothing "manual" about the first three from the perspective of a client (programmer). GC proponents complain about the work to create things like (1), (2), and (3) -- but there is no work for Ada programmers when you are using language capabilities or widely available libraries. Most people shouldn't be creating containers -- there's no point, you'll have a hard time doing better than the language-defined ones, and your time could be better used doing something else.

Stand-Alone or In-Compiler Provers

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 9 Apr 2015 18:26:53 -0500
Subject: Re: Languages don't matter. A mathematical refutation
Newsgroups: comp.lang.ada

[...]

In any event, I think the proof stuff has to be an intergral part of the compiler, because it seriously effects the code that gets generated. (If, after all, you can prove F(X) = 10 is True, you can replace F(X) with 10 appropriately. That can be huge win in runtime, especially in things like the preconditions of Ada.)

Conference Calendar

Dirk Craeynest

KU Leuven. Email: Dirk.Craeynest@cs.kuleuven.be

This is a list of European and large, worldwide events that may be of interest to the Ada community. Further information on items marked ♦ is available in the Forthcoming Events section of the Journal. Items in larger font denote events with specific Ada focus. Items marked with ☺ denote events with close relation to Ada.

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

2015

July 01-05 — 39th **Annual IEEE International Computer Software and Applications Conf**erence (COMPSAC'2015), Taichung, Taiwan. Event includes: symposium on Embedded & Cyber-Physical Environments; symposium on Software Engineering Technologies & Applications; symposium on Security, Privacy and Trust Computing; symposium on Novel Applications and Technology Advances in Computing; symposium on Computer Education and Learning Technologies; etc.

July 06-07 — 20th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2015), Vilnius, Lithuania.

☺ July 06-10 — 29th **European Conference on Object-Oriented Programming** (ECOOP'2015), Prague, Czech Republic. Topics include: all areas of object technology and related software development technologies, such as concurrent and parallel systems, distributed computing, programming environments, versioning, refactoring, software evolution, language definition and design, language implementation, compiler construction, design methods, design patterns, aspects, components, modularity, type systems, program analysis, specification, verification, security, real-time systems, etc.

- ☺ July 06 — 10th **Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems** (ICOOOLPS'2015}. Topics include: implementation of fundamental OO and OO-like features (e.g. inheritance, parametric types, memory management, objects, prototypes), runtime systems (e.g. compilers, linkers, virtual machines, garbage collectors), optimizations (e.g. static or dynamic analyses, adaptive virtual machines), resource constraints (e.g. time for real-time systems, space or low-power for embedded systems) and relevant choices and tradeoffs (e.g. constant time vs. non-constant time mechanisms, separate compilation vs. global compilation, dynamic loading vs. global linking, dynamic checking vs. proof-carrying code).

☺ July 07-10 — 27th **Euromicro Conference on Real-Time Systems** (ECRTS'2015), Lund, Sweden. Topics include: all aspects of real-time systems, such as embedded/RT systems design, scheduling design and analysis, WCET analysis, RT operating systems and middlewares, mixed criticality design & assurance, RT applications, tools and compilers for embedded systems, etc.

July 13-16 — 10th IEEE **International Conference on Global Software Engineering** (ICGSE'2015), Ciudad Real, Spain. Theme: "Solutions for distributed product development and maintenance". Topics include: software design and architecture for distributed development, strategic issues in distributed development, industrial offshoring and outsourcing experiences, tools and infrastructure support for distributed teams, methods and processes for global organizations, etc.

July 18-24 — 27th **International Conference on Computer Aided Verification** (CAV'2015), San Francisco, California, USA. Topics include: theory and practice of computer-aided formal analysis methods for hardware and software systems, algorithms and tools for verifying models and implementations, program analysis and software verification, verification methods for parallel and concurrent hardware/software systems, testing and run-time analysis based on verification technology, applications

and case studies in verification, verification in industrial practice, verification techniques for security, etc.

July 18-19 7th **Working Conference on Verified Software: Theories, Tools, and Experiments** (VSTTE'2015). Topics include: education, specification languages, specification/verification case-studies, software design methods, automatic code generation, verification tools (e.g., static analysis, dynamic analysis, model checking, theorem proving, satisfiability), tool integration, integrated verification environments, etc.

July 20-24 **Software Technologies: Applications and Foundations** (STAF'2015), L'Aquila, Italy. Successor of the TOOLS federated event. Topics include: practical and foundational advances in software technology, from object-oriented design, testing, mathematical approaches to modelling and verification, transformation, model-driven engineering, aspect-oriented techniques, and tools.

July 20-24 9th **International Conference on Tests And Proofs** (TAP'2015). Topics include: the synergy of proofs and tests, to the application of techniques from both sides and their combination for the advancement of software quality; transfer of concepts from testing to proving (e.g., coverage criteria) and from proving to testing; program proving with the aid of testing techniques; verification and testing techniques combining proofs and tests; generation of test data, oracles, or preambles by deductive techniques; automatic bug finding; case studies combining tests and proofs; formal frameworks; tool descriptions and experience reports; etc.

July 21-23 34th Annual ACM SIGACT-SIGOPS **Symposium on Principles of Distributed Computing** (PODC'2015), Donostia-San Sebastián, Spain.

August 03-05 IEEE **International Conference on Software Quality, Reliability and Security** (QRS'2015), Vancouver, Canada. Merger of SERE conference (International Conference on Software Security and Reliability) and QSIC conference (International Conference on Quality Software). Topics include: reliability, security, availability, and safety of software systems; software testing, verification and validation; software vulnerabilities; formal methods; benchmark, tools, and empirical studies; etc.

☺ August 20-22 13th IEEE **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2015), Helsinki, Finland. Topics include: parallel and distributed algorithms; tools/environments for parallel/distributed software development; novel parallel programming paradigms; code generation and optimization; compilers for parallel computers; middleware and tools; scheduling and resource management; reliability, fault tolerance, dependability, and security; parallel and distributed systems and architectures; applications of parallel and distributed processing; high-performance scientific and engineering computing; etc.

August 24-26 17th IEEE **International Conference on High Performance Computing and Communications** (HPCC'2015), New York, USA. Topics include: languages and compilers for high performance computing, parallel and distributed software technologies, parallel and distributed algorithms, embedded systems, tools and environments for software development, distributed systems and applications, high-performance scientific and engineering computing, reliability and fault-tolerance, trust, security, etc.

☺ August 24-28 21st **International European Conference on Parallel Computing** (Euro-Par'2015), Vienna, Austria. Topics include: all aspects of parallel and distributed processing, such as support tools and environments, scheduling, compilers, distributed systems and algorithms, parallel and distributed programming and languages, multicore and manycore programming, theory and algorithms for parallel computation, etc.

August 26-28 41st **Euromicro Conference on Software Engineering and Advanced Applications** (SEAA'2015), Madeira, Portugal. Topics include: information technology for software-intensive systems; model-based development, components and services (MOCS); software process and product improvement (SPPI); embedded software engineering (ESE); cyber-physical systems (CPS); etc.

Aug 31 – Sep 09 10th **Joint European Meeting of the Software Engineering Conference** and the ACM SIGSOFT **Symposium on the Foundations of Software Engineering** (ESEC/FSE'2015), Bergamo, Italy. Topics include: components and middleware, development environments and tools, distributed software, embedded and real-time software, maintenance and evolution, model-driven software engineering, parallel and concurrent software, reverse- and re-engineering, software architecture, software economics, validation, verification, and testing, etc.

☺ Sep 01-04 **International Conference on Parallel Computing 2015** (ParCo'2015), Edinburgh, Scotland, UK. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments, in particular parallel programming languages, compilers, and environments, tools and techniques for generating reliable and efficient parallel code, testing and debugging techniques and tools, best practices of parallel computing on multicore, manycore, and stream processors, etc.

☺ Sep 01-04 44th **Annual International Conference on Parallel Processing** (ICPP'2015), Beijing, China. Topics include: all aspects of parallel and distributed computing, such as applications, architectures, compilers, programming models, etc.

☺ Sep 01-04 **International Workshop on Embedded Multicore Systems** (EMS'2015). Topics include: programming models for embedded multicore systems; software for multicore, GPU, and embedded architectures; real-time system designs for embedded multicore environments; applications for automobile electronics of multicore designs; compiler for worst-case execution time analysis; formal method for embedded systems; etc.

September 01-04 15th **Workshop on Automated Verification of Critical Systems** (AVoCS'2015), Edinburgh, Scotland, UK. Topics include: model checking, specification and refinement, verification of software and hardware, specification and verification of fault tolerance and resilience, real-time systems, dependable systems, verified system development, industrial applications, etc. Deadline for submissions: August 7, 2015 (research ideas). Deadline for early registration: August 18, 2015.

September 06-09 11th **International Conference on Parallel Processing and Applied Mathematics** (PPAM'2015), Krakow, Poland. Topics include: multi-core and many-core parallel computing; parallel/distributed algorithms (numerical and non-numerical); scheduling, mapping, load balancing; parallel/distributed programming; tools and environments for parallel/distributed computing; security and dependability in parallel/distributed environments; applications of parallel/distributed computing; etc.

☺ Sep 06-09 6th **Workshop on Language-Based Parallel Programming Models** (WLPP'2015). Topics include: language and library implementations; proposals for, and evaluation of, language extensions; applications development experiences; comparisons between programming models; compiler implementation and optimization; etc.

September 07-08 7th **International Workshop on Software Engineering for Resilient Systems** (SERENE'2015), Paris, France. Topics include: requirements engineering & re-engineering for resilience; frameworks, patterns and software architectures for resilience; design of trustworthy systems; verification, validation and evaluation of resilience; empirical studies in the domain of resilient systems; methodologies adopted in industrial contexts; etc.

September 07-11 13th **International Conference on Software Engineering and Formal Methods** (SEFM'2015), York, UK. Topics include: abstraction and refinement; programming languages, program analysis and type theory; formal methods for real-time, hybrid and embedded/cyber-physical systems; formal methods for safety-critical, fault-tolerant and secure systems; software verification and validation; formal aspects of software evolution and maintenance; light-weight and scalable formal methods; tool integration; applications of formal methods, industrial case studies and technology transfer; education and formal methods; etc.

September 07-11 11th **European Dependable Computing Conference** (EDCC'2015), Paris, France. Topics include: theory, techniques, systems, and tools for the design, validation, operation and evaluation of dependable and secure computing systems, covering any fault model, from traditional hardware and software faults to accidental and malicious human interactions; dependability in practice (industrial applications, experience in introducing dependability in industry, use of new or mature dependability approaches to new challenging problems or domains, ...); hardware and software architectures of dependable and secure systems; safety critical systems; embedded and real-time systems; cyber-physical systems (e.g. networked embedded systems; automotive, aerospace, and medical systems); impact of manufacturing technology on dependability; verification and validation methods (e.g. testing and model checking); security of systems and networks; dependability and security in business and e-commerce applications; etc.

September 13-16 **Federated Conference on Computer Science and Information Systems** (FedCSIS'2015), Lodz, Poland.

☺ Sep 13-16 5th **Workshop on Advances in Programming Languages** (WAPL'2015). Topics include: compiling techniques; domain-specific languages; generative and generic programming; languages and tools for trustworthy computing; language concepts, design and implementation; model-driven engineering languages and systems; practical experiences with programming languages; program analysis, optimization and verification; programming tools and environments; specification languages; type systems; etc. Deadline for early registration: July 1, 2015.

Sep 13-16 8th **Workshop on Computer Aspects of Numerical Algorithms** (CANA'2015). Topics include: parallel numerical algorithms; libraries for numerical computations; languages, tools and environments for programming numerical algorithms; paradigms of programming numerical algorithms; etc.

September 15-17 14th **International Conference on Intelligent Software Methodologies, Tools and Techniques** (SoMeT'2015), Naples, Italy. Topics include: software methodologies and tools for robust, reliable, non-fragile software design; software developments techniques and legacy systems; software evolution techniques; agile software and lean methods; formal methods for software design; software maintenance; software security tools and techniques; 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.

September 22-25 15th **International Conference on Runtime Verification** (RV'2015), Vienna, Austria. Topics include: monitoring and analysis of software and hardware system executions. Application areas include: safety/mission-critical systems, enterprise and systems software, autonomous and reactive control systems, health management and diagnosis systems, and system security and privacy.

September 27-28 15th IEEE **International Working Conference on Source Code Analysis and Manipulation** (SCAM'2015), Bremen, Germany. Topics include: program transformation and refactoring, static and dynamic analysis, source level source metrics, security vulnerability analysis, source-level verification, program comprehension, bad smell detection, abstract interpretation, etc. Deadline for submissions: July 3, 2015 (tool papers).

September 27-30 15th **International Conference on Formal Methods in Computer-Aided Design** (FMCAD'2015), Austin, Texas, USA. Topics include: theory and application of formal methods in computer-aided design and verification of computer systems and related topics; synthesis and compilation for computer system descriptions, modeling, specification, and implementation languages; model-based design; correct-by-construction methods; experience with the application of formal and semi-formal methods to industrial-scale designs; etc.

Sep 28 – Oct 10 34th **International Symposium on Reliable Distributed Systems** (SRDS'2015), Montreal, Canada. Topics include: distributed objects and middleware systems, experimental or analytical evaluations of dependable distributed systems, formal methods and foundations for dependable distributed computing, high-assurance and safety-critical distributed system design and evaluation, secure and trusted distributed systems, dependability in cyberphysical systems, etc.

Sep 28 – Oct 10 24th **Australasian Software Engineering Conference** (ASWEC'2015), Adelaide, Australia. Theme: "Engineering Software for Innovation, Security, and Sustainability". Topics include: empirical research in software engineering; formal methods; large-scale distributed software engineering; legacy systems and software maintenance; model driven engineering; object and component-based software engineering; open source software development; programming languages; quality assurance; real-time and embedded software; software architecture; software design and patterns; software engineering education; software processes and quality; software re-use and product development; software reverse engineering; software risk management; software security, safety and reliability; software verification and validation; software vulnerabilities; standards; analysis and verification; etc. Deadline for submissions: July 3, 2015 (short research papers), July 26, 2015 (Doctoral Symposium papers).

Sep 29 – Oct 10 31st **International Conference on Software Maintenance and Evolution** (ICSME'2015), Bremen, Germany. Topics include: reverse engineering and re-engineering, software refactoring and restructuring, software migration and renovation, software and system comprehension, software repository analysis and mining, software testing, maintenance and evolution processes, software quality assessment, etc.

October 08 — 5th **International Workshop on Design, Modeling and Evaluation of Cyber Physical Systems** (CyPhy'2015), Amsterdam, the Netherlands. Topics include: development of industrial or research-oriented cyber-physical systems in domains such as robotics, smart systems (homes, vehicles, buildings), medical and healthcare devices, future generation networks; comparisons of state of the art tools in industrial practice; etc.

October 12-14 — 17th **International System Design Languages Forum** (SDL'2015), Berlin, Germany. Topics include: industrial application reports (industrial usage reports, standardization activities, tool support and frameworks, domain-specific applicability such as telecommunications, aerospace, automotive, control, ...), model-driven development, evolution of development languages (domain-specific language profiles especially for dependability, modular language design, semantics and evaluation, methodology for application, ...), etc.

October 12-15 — 13th **International Symposium on Automated Technology for Verification and Analysis** (ATVA'2015), Shanghai, China. Topics include: program analysis and software verification; analytical techniques for safety, security, and dependability; testing and runtime analysis based on verification technology; analysis and verification of parallel and concurrent hardware/software systems; verification in industrial practice; applications and case studies; etc.

October 12-15 — 27th **Annual IEEE Software Technology Conference** (STC'2015), Long Beach, California, USA. Topics include: critical infrastructure challenges, agile/lean development, affordability, open source, systems engineering challenges for software-intensive systems, etc.

☺ Oct 18-21 — 24th **International Conference on Parallel Architectures and Compilation Techniques** (PACT'2015), San Francisco, California, USA. Topics include: parallel architectures and computational models; compilers and tools for parallel computer systems; middleware and run time system support for parallel computing; support for correctness in concurrent hardware and software; parallel programming languages, algorithms and applications; applications and experimental systems studies; etc. Deadline for submissions: August 10, 2015 (ACM Student Research Competition).

October 21-23 — 18th IEEE **International Conference on Computational Science and Engineering** (CSE'2015), Porto, Portugal. Includes tracks on: scientific and engineering computing; CSE education; embedded and ubiquitous computing; security, privacy and trust; distributed and parallel computing; dependable, reliable and autonomic computing; etc.

- ☺ Oct 21 — **Workshop on Exascale Multi/many Core Computing Systems** (MuCoCoS'2015). Topics include: methods and tools for preparing applications for exascale; programming models, languages, libraries and compilation techniques; run-time systems; etc. Deadline for registration: September 4, 2015.

October 25-27 — ACM SIGPLAN **8th International Conference on Software Language Engineering** (SLE'2015), Pittsburgh, Pennsylvania, USA. Topics include: techniques for software language reuse, evolution and management of variations (syntactic/semantic) within language families; applications of DSLs for different purposes (incl. modeling, simulating, generation, description, checking); novel applications and/or empirical studies on any aspect of SLE (development, use, deployment, and maintenance of software languages); etc.

☺ Oct 25-30 — ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2015), Pittsburgh, Pennsylvania, USA. Topics include: all aspects of software construction and delivery, at the intersection of programming, languages, and software engineering. Deadline for submissions: August 7, 2015 (student volunteers). Deadline for early registration: September 25, 2015.

November 02-05 — 26th IEEE **International Symposium on Software Reliability Engineering** (ISSRE'2015), Washington DC, USA. Topics include: reliability, availability, and safety of software systems; verification and validation; software quality; software security; dependability, fault tolerance, survivability, and resilience of software systems; systems (hardware + software) reliability engineering; etc.

November 03-06 — 17th **International Conference on Formal Engineering Methods** (ICFEM'2015), Paris, France. Topics include: abstraction and refinement; program analysis; software verification; software model checking; formal methods for object and component systems, concurrent and real-time systems, cyber-physical systems, for software safety, security, reliability and dependability; tool development,

integration and experiments involving verified systems; formal methods used in certifying products under international standards; formal model-based development and code generation; etc.

November 04-06 **Symposium on Dependable Software Engineering: Theories, Tools and Applications** (SETTA'2015), Nanjing, China. Topics include: formalisms for modeling, design and implementation; model checking, theorem proving, and decision procedures; scalable approaches to formal system analysis; integration of formal methods into software engineering practice; contract-based engineering of components, systems, and systems of systems; formal and engineering aspects of software evolution and maintenance; parallel and multicore programming; embedded, real-time, hybrid, and cyber-physical systems; mixed-critical applications and systems; safety, reliability, robustness, and fault-tolerance; applications and industrial experience reports; tool integration; etc.

♦ November 05 **High Integrity Software 2015** (HIS'2015), Bristol, UK. Sponsored by AdaCore and Altran.

November 15-20 10th **International Conference on Software Engineering Advances** (ICSEA'2015), Barcelona, Spain. Topics include: advances in fundamentals for software development; advanced mechanisms for software development; advanced design tools for developing software; software security, privacy, safeness; specialized software advanced applications; open source software; agile software techniques; software deployment and maintenance; software engineering techniques, metrics, and formalisms; software economics, adoption, and education; improving productivity in research on software engineering; etc.

November 18-20 21st IEEE **Pacific Rim International Symposium on Dependable Computing** (PRDC'2015), Zhangjiajie, China. Topics include: software and hardware reliability, testing, verification, and validation; dependability measurement, modeling, evaluation, and tools; software aging and rejuvenation; safety-critical systems and software; dependability issues in distributed and parallel systems, in real-time systems, in aerospace and embedded systems, in cyber-physical systems, ...; etc. Deadline for submissions: July 22, 2015 (fast abstracts, industry track), August 20, 2015 (posters).

Nov 30 - Dec 12 13th **Asian Symposium on Programming Languages and Systems** (APLAS'2015), Pohang, Korea. Topics include: foundational and practical issues in programming languages and systems, such as semantics, design of languages and type systems, domain-specific languages, compilers, interpreters, abstract machines, program analysis, verification, model-checking, software security, concurrency and parallelism, tools and environments for programming and implementation, etc.

December 01-04 22nd **Asia-Pacific Software Engineering Conference** (APSEC'2015), New Delhi, India. Theme: "Software Process and Product Engineering". Topics include: embedded real-time systems; formal methods; product-line software engineering; SE environments and tools; security, reliability, and privacy; software architecture and design; software engineering methods; software maintenance and evolution; software process and standards; testing, verification, and validation; etc. Deadline for submissions: July 6, 2015 (regular research papers), July 31, 2015 (workshops, tutorials, post graduate symposium papers).

December 02-04 16th **International Conference on Product Focused Software Process Improvement** (PROFES'2015), Bolzano-Bozen, Italy. Topics include: software engineering techniques, methods, and technologies for product-focused software development and process improvement as well as their practical application in industrial settings.

December 08-11 16th ACM/IFIP/USENIX **International Middleware Conference** (Middleware'2015), Vancouver, Canada. Topics include: design, implementation, deployment, and evaluation of distributed system platforms and architectures for computing, storage, and communication environments; reliability and fault-tolerance; real-time solutions; scalability and performance; programming frameworks, parallel programming, and design methodologies for middleware; methodologies and tools for middleware design, implementation, verification, and evaluation; retrospective reviews of middleware paradigms; etc.

December 09-12 20th **International Conference on Engineering of Complex Computer Systems** (ICECCS'2015), Gold Coast, Australia. Topics include: verification and validation, security and privacy of complex systems, model-driven development, reverse engineering and refactoring, design by contract, agile methods, safety-critical & fault-tolerant architectures, real-time and embedded systems, cyber-physical systems, tools and tool integration, past reflections and future outlooks, industrial case studies, etc. Deadline for submissions: July 5, 2015 (workshops).

December 10 200th birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

2016

January 07-09 — 17th IEEE **International Symposium on High Assurance Systems Engineering** (HASE'2016), Orlando, Florida, USA. Topics include: tools and techniques used to design and construct systems that, in addition to meeting their functional objectives, are safe, secure, and reliable. Deadline for submissions: September 1, 2015 (papers).

January 19-22 — 8th **Software Quality Days Conference** (SWQD'2016), Vienna, Austria. Theme: "The Future of Systems and Software Development: Build in Quality & Efficiency right from the Start". Topics include: improvement of software development methods and processes; testing and quality assurance of software and software-intensive systems; domain specific quality issues such as embedded, medical, automotive systems; novel trends in software quality; etc.

April 02-08 — 19th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2016), Eindhoven, the Netherlands. Events include: ESOP (European Symposium on Programming), FASE, Fundamental Approaches to Software Engineering), FOSSACS (Foundations of Software Science and Computation Structures), POST (Principles of Security and Trust), TACAS (Tools and Algorithms for the Construction and Analysis of Systems).

June 01-05 — 12th **International Conference on integrated Formal Methods** (iFM'2016), Reykjavík, Iceland. Topics include: hybrid approaches to formal modelling and analysis; i.e., the combination of (formal and semi-formal) methods for system development, regarding modelling and analysis, and covering all aspects from language design through verification and analysis techniques to tools and their integration into software engineering practice. Deadline for submissions: July 20, 2015 (workshops).

♦ June 13-17 — 21st **International Conference on Reliable Software Technologies - Ada-Europe'2016** Pisa, Italy. Sponsored by Ada-Europe, in cooperation (pending) with ACM SIGAda, SIGBED, SIGPLAN, and the Ada Resource Association (ARA). Deadline for submissions: January 17, 2016 (papers, tutorials, workshops, industrial presentations).

December 10 — Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

The second UK conference on High Integrity Software will take place in Bristol, UK, on 5th November 2015. This one-day event offers the UK's foremost opportunity for engineers to share information about challenges and solutions in the domain of trustworthy software engineering for safety, security and business-critical applications.

This year's conference will feature three keynote speakers. Prof. Ian Phillips, Principal Staff Engineer at ARM, will talk about the role of software in overall system integrity. Prof. Phil Koopman, CMU, will present a study of the Unintended Acceleration (UA) of Toyota vehicles and related software safety issues based on his role as an expert witness. Prof. Mark Little, Vice President Red Hat and CTO of JBoss, will talk about the success of open source software in mission-critical environments and its future role in innovative areas including the Internet of Things.

The programme will also feature technical sessions on software safety, tools & architectures, and threats & security. More details are available on the conference website.

The event includes an exhibition at which vendors will be presenting their tools and services offer for the high integrity software domain. The exhibition will be open during the morning and afternoon breaks, during lunchtime and also during the networking "cocktail hour" at the end of the day.

Attendance at HIS 2015 will cost £175 per delegate, which covers all aspects of this event (breaks, lunches, sessions, exhibition and networking drinks afterwards). Further information and instructions on how to register can be found on the conference website.

www.his-2015.co.uk

SPONSORED BY

AdaCore ALTRAN

Conference Chair

Giorgio Buttazzo
Scuola Superiore Sant'Anna

Program Co-Chairs

Marko Bertogna
Univ. of Modena and Reggio Emilia

Luís Miguel Pinho
CISTER Research Centre/ISEP

Special Session Chair

Eduardo Quiñones
Barcelona Supercomputing Center

Tutorial and Workshop Chair

Jorge Real
Universitat Politècnica de València

Industrial Co-Chairs

Marco Di Natale
Scuola Superiore Sant'Anna

Tullio Vardanega
Università di Padova

Publication Chair

Geoffrey Nelissen
CISTER Research Centre/ISEP

Exhibition Co-Chairs

Paolo Gai
Evidence Srl

Ahlan Marriot
White Elephant GmbH

Publicity Co-Chairs

Mauro Marinoni
Scuola Superiore Sant'Anna

Dirk Craeynest
Ada-Belgium & KU Leuven

Local Chair

Ettore Ricciardi
ISTI-CNR, Pisa

General Information

The **21st International Conference on Reliable Software Technologies – Ada-Europe 2016** will take place in Pisa, Italy. Following its traditional 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.

Schedule

17 January 2016	Submission of papers, industrial presentation, tutorial and workshop proposals.
10 March 2016	Notification of acceptance to all authors
24 March 2016	Camera-ready version of papers required
2 May 2016	Industrial presentations, tutorial and workshop material required

Topics

The conference has over the years become a leading international forum for providers, practitioners and researchers in reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the design, development and maintenance of long-lived, high-quality software systems for a challenging variety of application domains. The program will allow ample time for keynotes, Q&A sessions and discussions, and social events. Participants include practitioners and researchers representing industry, academia and government organizations active in the promotion and development of reliable software technologies.

This edition of Ada-Europe features a focused **Special Session on Safe, Predictable Parallel Software Technologies**. Following the increasing trend of usage of Multi-/Many-core systems, it is more and more important to assess how reliable software technologies need to adapt to these complex platforms, as well as how parallel models need to adapt to domains in which safety and predictability is a must. Topics include (but are not limited to): **Predictable Parallel Programming Models, Compiler Support for Parallel Execution, Parallel Runtimes, Automatic Parallelization, Safety Issues and Reliability Mechanisms for Parallel Execution, Software Modelling and Design Approaches**.

For the **general track of the conference**, topics of interest include but are not limited to (full list on the website): Real-Time and Embedded Systems, Mixed-Criticality Systems, Theory and Practice of High-Integrity Systems, Software Architectures, Methods and Techniques for Software Development and Maintenance, Software Quality, Mainstream and Emerging Applications, Experience Reports in Reliable System Development, Experiences with Ada.

Call for Regular and Special Session Papers

Authors of papers which are to undergo peer review for acceptance are invited to submit original contributions by 17 January 2016. Paper submissions shall not exceed 14 LNCS-style pages in length. Authors for both the general track and the special session shall submit their work via EasyChair at https://easychair.org/conferences/?conf=adaeurope2016. The format for submission is solely PDF.

Proceedings

The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer, and will be available at the conference. The authors of accepted regular and special session papers shall prepare camera-ready submissions in full conformance with the LNCS style, not exceeding 14 pages and strictly by 24 March 2016. 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 International Conference on Reliable Software Technologies is ranked class A in the CORE ranking and Microsoft Academic Search has it in the top third for conferences on programming languages. The conference is listed in DBLP, SCOPUS and Web of Science Conference Proceedings Citation index, among others.

Awards

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

Call for Industrial Presentations

The conference seeks industrial presentations which deliver value and insight but may not fit the selection process for regular papers. Authors are invited to submit a presentation outline of exactly 1 page in length by 17 January 2016. Submissions shall be made via EasyChair following the link https://easychair.org/conferences/?conf=adaeurope2016. The format for submission is solely PDF.

The Industrial Committee will review the submissions and make the selection. The authors of selected presentations shall prepare a final short abstract and submit it by 2 May 2016, aiming at a 20-minute talk. The authors of accepted presentations will be invited to submit corresponding articles for publication in the *Ada User Journal* (http://www.ada-europe.org/auj/), which will host the proceedings of the Industrial Program of the Conference. For any further information please contact the Industrial Co-chairs 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. 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 5; for half-day tutorials, these benefits will be accordingly halved. The *Ada User Journal* (http://www.ada-europe.org/auj/) 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 Tutorial and Workshop Chair. The workshop organizer shall also commit to preparing proceedings for timely publication in the *Ada User Journal* (http://www.ada-europe.org/auj/).

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.

Venue

The conference will take place at Scuola Superiore Sant'Anna (left images, including the aula magna where the main conference sessions will take place), in the heart of Pisa, Italy. June is full of events in Pisa, including in the conference week the Saint Patron's festivities (San Ranieri) with the Luminara on the night of June 16 (thousands of candles burn and reflect on the river – image on the right). Plan in advance! It is absolutely worth it!

Proven Test Solutions for Reliable Embedded Software

"VectorCAST is unique in that it provides us with the ability to increase the reliability and quality of our flight software."

-- Honeywell

VECTOR software

vectorcast.com

VectorCAST is a TÜV SÜD Certified Software Tool for Safety Related Development

Vector Software, Inc.
Golden Cross House | 8 Duncannon Street | London WC2N 4JF UK | +44 203 603 0120 | sales@vectorcast.com

Lovelace & Babbage and the Creation of the 1843 'Notes'*

John Fuegi and Jo Francis

Abstract

Augusta Ada Lovelace worked with Charles Babbage to create a description of Babbage's unbuilt invention, the Analytical Engine, a highly advanced mechanical calculator often considered a forerunner of the electronic calculating computers of the 20th century. Ada Lovelace's "Notes," describing the Analytical Engine, published in Taylor's Scientific Memoirs *in 1843, contained a ground-breaking description of the possibilities of programming the machine to go beyond number-crunching to "computing" in the wider sense in which we understand the term today. This article expands on research first presented by the authors in their documentary film,* To Dream Tomorrow.

Augusta Ada Lovelace in a portrait by Margaret Carpenter.
Photo by Jo Francis. Still image from *To Dream Tomorrow*, © Flare Productions, 2003/2015. Used with permission.

> What shall we do to get rid of Mr. Babbage and his calculating Machine? Surely if completed it would be worthless as far as science is concerned?
>
> --British Prime Minister Sir Robert Peel, 1842 [1]
>
> The Analytical Engine does not occupy common ground with mere 'calculating machines.' In enabling mechanism to combine together general symbols, in successions of unlimited variety and extent, a uniting link is established between the operations of matter and the abstract mental processes of the most abstract branch of mathematical science. A new, a vast and powerful language is developed for the future use of analysis.
>
> --A.A. Lovelace, "Notes by A.A.L.," 1843 [2]

Charles Babbage's Difference Engine and Analytical Engine, conceived in the first half of the 19th century, are often seen as anticipating key design features used in modern computing, even though none of Babbage's extraordinary devices was fully built in his lifetime. Augusta Ada Lovelace, née Byron, who worked against the restrictions on women of her day to successfully train as a mathematician, worked closely with Babbage to describe the more advanced of his engines, the Analytical Engine, in a collection of "Notes" published in *Taylor's Scientific Memoirs* in 1843. Lovelace's vision of the Engines' potential for the future of computation may now be seen as having exceeded Babbage's own vision for his machines in several key ways. She became the first person known to have crossed the intellectual threshold between conceptualizing computing as only for calculation on the one hand, and on the other hand, computing as we know it today: with wider applications made possible by symbolic substitution.

In an early background interview at the Science Museum (London) for the historical documentary film about collaboration between Lovelace and Babbage, *To Dream Tomorrow* [3], Babbage authority Doron Swade mentioned that he thought Babbage and Lovelace had "very different qualities of mind." Swade's observation proved to be of enormous value for our subsequent research.

An examination of the original Lovelace and Babbage documents shows that, whereas Babbage concentrated on the number-crunching possibilities of his new designs, Lovelace went beyond number-crunching to see possibilities for wider applications. She wrote:

> Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent [4].

Aware that the punched card mechanism guiding the decision list of the Analytical Engine was taken by Babbage from the Jacquard loom and that Jacquard had created pictures of great complexity by this means, she noted: "We may say most aptly, that the Analytical Engine *weaves algebraical patterns* just as the Jacquard-loom weaves flowers and leaves" [5]. Making her own

independence of thought clear within the "Notes," she wrote:

> Whether the inventor of this Engine had any such views in his mind while working on the invention, or whether he may subsequently ever have regarded it under this phase, we do not know; but it is one that forcibly occurred to ourselves [6] on becoming acquainted with the means through which analytical combinations are actually attained by the mechanism.[4]

In order for us to look closely at the original Lovelace and Babbage documents written at the time the "Notes" were being created, we had to go to a number of different archives. We also had to take care when examining most published accounts. Most extant books tended to be either primarily accounts of Lovelace with Babbage as an important but subsidiary figure, or accounts of Babbage with Lovelace often reduced to a largely marginal figure. In contrast, it was our intention in making *To Dream Tomorrow* to examine and acknowledge what each one did as an individual, as well as what the two achieved working together.

Since the "Notes" are the single most comprehensive description of the more advanced capabilities of the Analytical Engine and since a full-scale Analytical Engine was never built, the "Notes" constitute the main conduit through which Babbage's extraordinarily advanced engineering ideas influenced future generations. Consequently, the "Notes" and Lovelace's role in creating them, and the question of the extent to which she went beyond the ideas of Babbage are of historical significance.

Lovelace's letters to Babbage, with a large array of other vital Babbage materials, are held at the British Library in London. A large number of Babbage's drawings and notes, used by Doron Swade and the late Allan Bromley to reconstruct plans for Babbage's various "Engines" (some of the plans of which have been published in the *IEEE Annals*) are at London's Science Museum. A number of Babbage's letters to Lovelace are in the Byron/Lovelace collection at Oxford's Bodleian Library. Lovelace Estate Records (the documents showing the financial and other material conditions under which Ada worked after she married in 1835), are held at the County Historical Archive in Woking, UK. As these historic materials have never been published in their entirety, their interrelationship has often remained almost entirely unexamined.

Over the last four years, to gain access to and to use Augusta Ada Lovelace materials, we needed to obtain the permission of Ada's great-great-grandson, the current Earl of Lytton. Lord Lytton was pleased at, among other things, the idea of taking a fresh look at the role in Ada's education of his great-great-great-grandmother, Lady Byron. Ada grew up essentially in a single-parent home; Lady Byron left the abusive household of the famous poet Lord Byron when Ada was five weeks old. Lady Byron (who had herself received some training in mathematics) was primarily responsible for Ada's education up to and including the time Ada met Charles Babbage when she was 17 and he 42, and she first saw Babbage's prototype Difference Engine, a mechanical calculator.

It is important to note what happens both for Ada and for Charles Babbage in the 10 years that lie between Ada Byron's first view of the prototype Difference Engine in 1833 and the creation of the "Notes" in 1843. In this period, the ideas of Babbage undergirding the more advanced calculating device, the Analytical Engine, emerged. Ada was present as the key new ideas were discussed between Charles Babbage and the great science expositor, Mary Somerville. By 1834, both Somerville and Babbage were mentors for then 18-year-old Ada, and Babbage supplied Ada with a number of engineering drawings so she could better understand his newest designs.

Though Ada Byron (like her mother before her) was barred, as a woman, from attending university in England at that time, she worked with a series of tutors in mathematics. After meeting Babbage, her mathematical studies began to focus on what she needed to know to advance her understanding of the principles behind Babbage's Difference and Analytical Engines. Her study advanced even after she married William, soon-to-be-named Earl of Lovelace, and had three children in a little over three years; the last born in July 1839. In 1840 she began a series of tutorials with Augustus De Morgan, professor of mathematics at University College, London.

Babbage had first received a grant from the British government in 1823 to begin to build a Difference Engine. Yet, despite expending large sums of public funds and a great deal of his own money, by 1833 he had failed to complete more than a small prototype Difference Engine. This prototype is a fully functioning device that can be seen today at the Science Museum in London. By 1834, however, Babbage began talking about having an even more complex undertaking to displace the earlier one. As Ada, over a span of a decade, extended her capacities for understanding Babbage's Engines, in the same period, Babbage himself felt frustrated by being unable to convince British authorities of the importance of his latest design, a proposal for an Analytical Engine of vastly greater scope than his earlier Difference Engine. But, by now, the British government was frustrated by almost two decades of dealings with Babbage. On 11 November 1842 the inventor had a meeting with the Prime Minister, Sir Robert Peel. Babbage (as we know from his own notes [7]) spent more time attacking the government than describing the new Engine. Peel, for his part, had, prior to the meeting, solicited support to call Babbage's work officially worthless. The meeting was a disaster. Both men talked past one another. On January 5, 1843, Babbage was informed the government had sent the prototype of the Difference Engine to the King's College Museum [8]. In March, Peel formally withdrew support for the project, and only a single voice in parliament was raised on Babbage's behalf. By 1843 it was clear that Babbage, for all his technical brilliance, had been rejected in England for further government funding for completing either the older Difference Engine or the newer Analytical Engine.

Before the formal rejection by Peel in 1842-1843, Babbage had gone to Turin in the fall of 1840 hoping to line up foreign support for his plans. Before going to Turin he had had printed the 24-in. by 36-in. "Plan, #25," one version of the ever-changing Analytical Engine design. In Turin, a young engineer, Luigi Menabrea, took notes on Babbage's talks and began to prepare an article based on what Babbage presented. Menabrea's article, "Notions sur la machine analytique," was published in the journal *Bibliothèque Universelle de Genève*, in October 1842.

When copies of the Menabrea article reached England in the fall of 1842 and Babbage had had his disastrous meeting with Peel, the French language article was discussed by Ada Lovelace and the inventor, Charles Wheatstone. Both Lovelace and Wheatstone were probably better informed about the Difference and Analytical Engines than anyone other than Babbage himself and possibly Somerville, and they had more knowledge than Menabrea, who had met Babbage only briefly in Turin. Wheatstone, a close friend of Babbage and Lovelace, was one of the best informed people in Britain on developing and marketing new technologies. By 1837, the Cooke/Wheatstone Telegraph had been patented, financed, built and marketed with a highly successful advertising campaign promoting the device. Wheatstone had also worked on designs for calculating by machinery as we know from an 18 May 1839 entry in Babbage's Notebook: "Yesterday saw Wheatstone's model for telegraph and his drawings for Multiplication Engine." According to Anthony Hyman who cites the Babbage Notebook, "Wheatstone's apparatus gave Babbage the idea that he might use electro-mechanical switching instead of mechanical techniques for the Calculating Engines. "[9]

Considering the date, 1839, the idea is breathtaking, coming almost a century ahead of Howard Aiken making his first advanced calculator proposals to IBM [10]. Even though Babbage had not adopted Wheatstone's electro-mechanical switching in 1839, in 1937 Aiken directly mentioned Babbage's engine designs as a precursor and joked, "If Babbage had lived 75 years later I would have been out of a job." [11] Descriptions of Babbage's designs were also turned up by Konrad Zuse in Berlin as part of his "prior art" patent search in 1937, and similar references crop up as well in accounts of the work of John von Neumann. H. J. Gray notes: "John von Neumann urged that all the machine units be connected . . . so that the machine could be used as a computer of the Babbage type. . . This was done and ENIAC was operated in this fashion until it retired." [12] A further link is a reported conversation of John von Neumann with S. Frankel cited by Andrew Hodges [13]. Hodges also notes that Turing was aware both of Babbage and Lovelace [14]. Thus some links can be shown between key 20th century figures in computer history (Turing, Aiken, von Neumann), and the work done in England in the early 1840s, but dismissed by the British government then as worthless.

In the fall of 1842, aware of what had happened between Babbage and Peel, Wheatstone and Lovelace, not yet mentioning the idea to Babbage as he was ill after his meeting with Peel, thought it could help the cause of advancing Babbage's work in England if Ada would translate the Menabrea article into English. She was skilled in French, as her mother had arranged for her to study languages from childhood on and encouraged her to polish her skills during a 15-month period they had spent abroad. Lovelace went ahead with the translation over the winter of 1842-1843.

In early 1843, Lovelace showed Babbage what she had been working on over the winter. Babbage's response shows the high regard in which he held Lovelace's intellect and her understanding of his work. Babbage recollected 20 years later:

> Some time after the appearance of [Menabrea's] memoir the late Countess of Lovelace informed me that she had translated the memoir of Menabrea. I asked why she had not herself written an original paper on a subject with which she was so intimately acquainted? To this Lady Lovelace replied that the thought had not occurred to her. I then suggested that she should add some notes to Menabrea's memoir; an idea that was immediately adopted." [15]

The resulting "Notes" are three times the length of Menabrea's essay and contain the most influential insights.

Lovelace (as we can confirm from her letters held at the British Library), wrote the "Notes" mainly at Ockham Park, an hour south of London. Babbage wrote back to her from his Dorset Street house in London, adjacent to his custom-built, fireproof workshop. They met together to discuss problems and to do proof-reading at Ada's London house, 12 St. James's Square. Records in Lovelace's, Babbage's, and Wheatstone's handwriting at the British Library and at the Bodleian Library allow us to follow in an almost hourly way how the "Notes" came into being over the summer of 1843. With multiple mail deliveries each day, and with more missives delivered by personal messenger, one gets a sense of the mutual excitement, collegiality, but sometimes fierce frustration on both sides of the exchange. The letters crossed and recrossed as Lovelace's working days sometimes stretched to 18 hours.

One remarkable feature of Lovelace's "Notes" is that they describe not the physical reality of a single existing Analytical Engine but what historian of technology Sadie Plant has called "a virtual machine." "It is virtual on two levels," said Plant when interviewed for *To Dream Tomorrow*. "She is," notes Plant, "writing the programs for a virtual machine, for a future machine in effect." Most of the mechanical parts for the Engine did not yet exist, and the drawings, even when they did exist and Babbage could put his hands on them, were incompatible as they reflected different stages of design over a nine-year period. We know that, even at the last stage, as the "Notes" were in press, Babbage told Lovelace on 18 August 1843:

> My Dear Lady Lovelace I much fear the drawings will not be very intelligible. They were never published and

only a few proofs were taken. I will endeavour to find a complete set and bring them with me on Monday. [16]

To create a comprehensive description of the Analytical Engine that did not (and indeed does not) exist, a machine that was in a constant state of flux in Lovelace's and Babbage's lifetime and for which Babbage had difficulty turning up a full, internally consistent set of drawings--was to attempt something of almost inconceivable difficulty. Swade reports in *The Cogwheel Brain* how immensely difficult it was for him and Allan Bromley, even over a period of several years, to work through thousands of pages of Babbage's "Notes" in order to understand a vast, unbuilt, constantly changing entity. Groping to arrive at correct formulations during a single intense summer of work in 1843, Lovelace and Babbage exchanged letters that are startlingly modern, almost email-like: abrupt, often informal, dashed off, and sent with uncorrected errors.

Lovelace in the summer of 1843 was 27 years old and saw herself, as she noted in a letter to a relative, as "a fully professional person." [17] Her letters to Babbage mix respect with banter, and sometimes the bluntest frankness when he loses papers or fails to remain focused on the task at hand. Tellingly, she often wrote "My Dear Babbage," using the form of male-to-male, colleague-to-colleague address of the Victorian era. Babbage, who was in 1843 in his early fifties, addressed her as 'My Dear Lady Lovelace."

The following letters give us a sense from Babbage's perspective of how the work was proceeding. Babbage, from Dorset St. 30 June 1843, writes to Lovelace at Ockham Park in such a hurry that not enough postage was put on the letter, so it is marked on the envelope "More To Pay."

> My Dear Lady Lovelace
>
> I am delighted with Note D. It is in your usual clear style and requires only one triffling [sic] alteration which I will make. This arises from our not having yet had time to examine the outline of the mechanical part. …
>
> I enclose a copy of the integration. I am still working at some most entangled notations of Division but see my way through them at the expense of heavy labour, from which I shall not shrink as long as my head can bear it. I have been somewhat impeded for the last few days. Your latest information was the most agreeable. Ever my dear Lady Lovelace Sincerely yours C. Babbage. [18]

On Sunday, 2 July 1843 Babbage wrote:

> I am very reluctant to return the admirable and philosophic view of the Abral. [sic] Engine contained in Note A. Pray do not alter it and do let me have it returned on Monday. I send also the rest of Note D. There is still one triffling [sic] misapprehension about the Variable cards--A Variable card may order any number of Variables to receive the same number upon theirs at the same instant of time--But a Variable card never can be directed to order more than one Variable to be given off at once because the mill could not receive it and the mechanism would not permit it. All this was impossible for you to know by intuition and the more I read your Notes the more surprised I am at them and regret not having earlier explored so rich a vein of the noblest metal.
>
> The account of them stands thus
>
> | | |
> |---|---|
> | A sent to Lady L. | F Retained by Lady L. |
> | B with CB | G Where is it gone?? |
> | C Ditto | H With CB |
> | D Sent to Lady L. | |
> | E With CB | |
>
> I have not seen Mr. Wheatstone and am ashamed to write until I can positively put the *whole* of the Notes into his hands.
>
> I will attend your commands tomorrow And am ever most truly yours C. Babbage [19]

Lovelace wrote back at once. She had decided that, since Babbage had made a mistake about how she viewed the variable cards, she would need to see him the next day in London to get several points clarified. She playfully and tactfully points out that in fact no Note C had ever existed.

> Ockham, Sunday 6 o'clock. I have worked incessantly and most successfully all day. You will admire the *Table and Diagram* extremely.
>
> They have been made out with extreme care and all the indices most minutely and scrupulously attended to. Lord L[ovelace] is at this moment kindly *inking it all over* for me. I had to do it in pencil.
>
> You must bring *all* the Notes with you tomorrow as I have observations to make on each one and especially on this final one H.
>
> There never was a Note C. I do not know why I chose H instead of C and thus insulted the latter worthy letter.
>
> I cannot imagine what you mean about the Variable-Cards; since I never either supposed in my own mind that one Variable-card *could give off* more than one Variable at a time; nor have (as far as I can make out) expressed such an idea in any passage whatsoever …[20]

Having met with Babbage in London to work through the observations each had made, Lovelace wrote to him, both to clarify the issue of the variable cards and to convey her decision to assert her authorship of the "Notes":

> Ockham, Tuesday Morning … Lord L. suggests my signing the translation and the Notes, by which he means simply putting … "*translated by A.A.L.;*" & adding to each Note the initials A.A.L.
>
> It is not my wish to *proclaim* who has written it; at the same time that I rather wish to appear anything that may tend hereafter to *individualize* and *identify* it with other productions of the said A.A.L.

> My third topic, tho' my last is my most anxious and important. I have yesterday evening and this morning very amply analyzed the question of the *number* of Variable Cards, as mentioned in the final Note H (or G?). And I find that you and I between us have made a *mess* of it; (for which I can perfectly account in a very natural manner). I enclose what I wish to inscribe *instead* of that which is now there. I think the present wrong passage is only about eight or ten lines, & is I believe on the *second* of the three great sheets which are to *follow* the diagram.
>
> The fact is that if my own composition about the Variable Cards in *Note D* had been strictly followed by myself in Note H this error would not have occurred. The confusion has arisen simply from the circumstance of applying to the *Variable* Cards, facts which relate to the *Operation*-Cards. In Note D it is very well and lucidly demonstrated that every simple operation demands the use of at least those Variable Cards. It does not signify whether the operations be in *cycles* or not. A million successive additions would each demand the use of these new Variable Cards under ordinary circumstances. In Note H, the erroneous lines are founded on the hasty supposition that the cycle or recurring group of *Operation*-Cards (13 . . . . .23) will be fed by a cycle or recurring group of *Variable*-Cards.
>
> I enclose what I believe it ought to be. If already gone to the printer we must alter that passage in the proofs unless you could call at the printers and there paste over the amendment. [21]

She commented further on the technical issues in another letter to Babbage, probably also of 4 July. This letter is dated only "Tuesday 1843," but the context makes plain that it was written near the same time:

> My Dear Babbage.
>
> I hope you will approve of what I send. I have taken *much* pains with it. I have explained that there would be, in this instance & in many others, a recurring group or cycle of *Variable* as well as of *Operation* Cards; and I have (I think very judiciously and easily) touched on the only departures from *perfect* identity which *could* exist during the repetitions of (13 . . . . . 23); and yet have not *committed* myself by saying if the departures would require to be met by the introduction of one or more *new* cards or not; but have simply indicated that as the associations follow a regular rule, they would be easily provided for. I think I have done it admirably and diplomatically (*Here* comes in the *intrigante* and *politician*!) Ever yours A.L. [22]

Lovelace's Note G describes how the Analytical Engine could be used to calculate the values of the Bernoulli numbers. Lovelace, knowing that Babbage believed the Engine could have the capacity to handle Bernoulli numbers, as he had discussed in a letter circa January 1841 to the German savant Alexander von Humboldt [23], took it upon herself to make sure there was a written description and demonstration of how this could be done. She writes from Ockham Park on Wednesday, 5 July 1843:

> I do not go to town until Monday. Keep yourself open if you can for that day in case there is anything I wish to see you about which is very likely. But the *evening* I think is most likely to be my time for you, as I rather expect to be engaged incessantly until after 6 o'clock. I shall sleep in town that night.
>
> I am doggedly attacking and sifting to the very bottom all the ways of deducing the Bernoulli Numbers. In the manner I am grappling with this subject; and connecting it with the others, I shall be some days upon it. …
>
> "Labore ipse voluptus" [Labor Is Its Own Reward] is in very deed my motto! And (as I hinted just now), it is perhaps well for this world that my line and inclination is more the *spiritual*; and that I have not taken it into my head or lived in times or circumstances calculated to put into my head to deal with the sword, poison, and intrigue in the place of x, y, & z. … [24]

In the archive, this letter is followed in folio 354 by a very brief note from Babbage dated Wednesday, July 1843, presumably of 5 July: "Return sheet with two corrections. Right about Card requiring new Variable." [25]

This was typical of a staccato to and fro. Ada, writing the "Notes," queried Babbage, as the inventor of the yet unbuilt Engine, as to whether or not he anticipated his Engine could do something and do it as she understood it. Babbage's replies suggest that he had learned something new about his own machine from Lovelace's queries and speculations. For instance, a letter headed "Ockham Thurs. Morn. 1843" reminds us that Lovelace was attempting a description of what Babbage himself was still in the process of clarifying. She wrote:

> My Dear Babbage. I have read your papers over with great attention. But I want you to answer me the following question by return post. The day I called on you, you wrote off on a scrap of paper (which I have unluckily lost); that the *Difference* Engine would do [Authors' note: Lovelace draws a small triangle here] (something or other) [Authors' note: The parentheses are hers] but that the *Analytical* Engine would do [Authors' note: Lovelace again draws a triangle here] (something else that is absolutely general). Be kind enough to write this out properly for me; and then I think I can make some *very* good Notes. … [26]

In another letter early in the process Lovelace had written

> My Dear Babbage. … I want to put in something about Bernoulli's Numbers in one of my Notes as an example of how an implicit function may be worked out by the engine without having been worked out by human head or hands first. Give me the necessary data and formulae. Yours ever AAL [27]

The correspondence brings to life the actual process of editing and proofreading:

> July 1843 Ockham Tuesday Morning. My Dear Babbage. … What I want to know is this: can you be with me in town at 4 o'clock. This is in order that I may *read over* aloud with you all the Notes. … [28]

The fact Lovelace wanted to go through the "Notes" with Babbage, and had previously sent him her translation of Menabrea to check makes it clear that proofreading was a joint undertaking, supplemented in the customary way by the printers. Given this fact, it seems odd to dismiss (as one severe critic has done) [29] only Lovelace for failing to catch an error made by the Swiss printer (an error of "cas" for "cos." uncaught by Menabrea), and then using this to claim Lovelace knew little about mathematics.

By the end of July, Lovelace and Babbage appeared to be on the final lap. Lovelace, the mother of three children with the Earl of Lovelace, jokingly wrote about the "Notes" as though they were her first child:

> Ockham Thursday morning 27 July: My Dear Babbage. …To say the truth I am rather amazed at them [the Notes] & I have made Lord Lovelace laugh much by the dryness with which I remarked "Well. I am very much satisfied with this first child of mine. He is an extremely fine baby and will grow to be a *man* of the first magnitude and power." [30]

A meticulous worker, Lovelace struggled not only with the difficulty of the material but also with the errors of the printers and Babbage himself. She wrote to Babbage from St. James's Square:

> The beginning of Note G (by which I mean the Table & all that precedes it) never has been returned into my hands; a small part of the remainder was, but that I speedily gave you back, and there it is now printed. --
>
> The missing part *must* be either at your house or at the printer's; & it seems to me *very* unlikely that you should have retained it. So altogether I would wager almost anything that it is at the *office*; or that if lost, it has been lost *there*.
>
> At the same time, I have also fancied you were a little harum-scarum & inaccurate now & then about the exact *order & arrangement* of sheets, pages, & paragraphs & c. (witness that paragraph which you so carelessly *pasted* over!)
>
> I suppose I must set to work to write something better, if I can, as a substitute. The *same precisely* I could not recall. I think I should be able in a couple of days to do something. However I should be deucedly inclined to *swear at you*, I will allow.
>
> I desire my messenger to wait; as it is possible you may have something to communicate more agreeable.
>
> I go soon after seven. I believe I shall *not* be in Town myself on Monday as I expected. Yours A.L. [31]

"Ockham Sunday Afternoon" Lovelace writes:

> I am half beside myself with hurry and work. … I wish you were as accurate and as much to be relied on as I am myself. You might often *save* me much trouble if you were; whereas you in reality *add* to my trouble not infrequently and there is at any rate always the anxiety of *doubting* if you will not get me into a scrape even when you don't.
>
> By the way, I hope you do not take upon yourself to alter any of my corrections. I must *beg* you not. They all have some very sufficient reason. And you have made a pretty mess and confusion in one or two places (which I will show you sometime) where you have ventured on my *M.S.'s* to *insert* or *alter* a phrase or word and have utterly muddled the sense. . . . [32]

From Lovelace's letters, it is clear that she thought the intense working period was yielding the desired result: a strong, persuasive article describing the capabilities and functioning of the Analytical Engine, to generate interest and support for its construction. But by early August the tone of the exchanges is increasingly acerbic as Lovelace realizes that Babbage is trying to convince the printer to include one of his diatribes (which he was however unwilling to sign). Babbage wanted at the last minute to prevent the publication of the article unless he could fulminate at length in the same issue about the way he had been and was being treated by the government. But Lovelace overrode him and had the printer proceed as originally planned. A key Babbage letter does not appear to have survived as it is not at the Bodleian in the Lovelace/Byron Collection. His letter must have been written around the beginning of August 1843 because Ada Lovelace's letter of 6 August is clearly in response to something from him about her overruling him on going ahead with the article.

> My Dear Babbage. … On the *one point* of *not withdrawing* the translation & Notes from the Memoir, nor consenting to its *separate* publication, I was entirely and finally decided; as I think neither for *your advantage* nor my own, to do so; added to my opinion that it would under the circumstances be dishonorable and unjustifiable … Be assured that I am your best friend; but that I never *can* or *will* support you in acting on principles which I conceive to be not only wrong in themselves, but *suicidal*. [33]

In his reply of Tuesday, 8 August 1843, Babbage protested her decision, yet seemed to acknowledge her authority to make it:

> My Dear Lady Lovelace
>
> I leave the Ms and also the proofs of the Notes I recd. last night and promised to send this evening.
>
> I will write to Printer to say you will send them up by post direct to them.
>
> This direct communication will save time and there is very little time to spare for this Number *ought* to be out in the course of a few days.
>
> I have nothing to add at present except that you do me injustice in supposing I wished you to break any

> engagement with the Editor. I wished you to ask him to allow you to withdraw from it. Had the Editor been in England I believe he would at my request have inserted my defense or forborn to have printed the paper--As it stands I have done all I can at present to defend myself and having failed in the most important part shall make the best I can of the rest. Ever truly yours C. Babbage [34]

Babbage's supposition about the editor's wishes did not turn out to be true. The editor backed Lovelace, not Babbage. Opposition to Babbage's diatribe idea was unanimous. Neither Wheatstone nor Charles Lyell, the eminent geologist and mutual friend of Lovelace and Babbage, thought Babbage's interests would be served by yet another attack. Despite the advice of his closest friends, Babbage published his diatribe separately, in a different magazine, a few weeks later [35].

Whatever Babbage might decide to do, Lovelace keenly felt her own responsibility for this project. On Tuesday, 8 August, she wrote to her mother:

> I have been harassed and puzzled in a most perplexing manner by the conduct of Mr. Babbage … I am sorry to come to the conclusion that he is one of the most *impracticable*, *selfish*, and *intemperate* persons one can have to do with … But I am happy to find that W. [Author's note: "W." indicated William, her husband] & Wheatstone entirely approves my conduct and means. I declared at once to Babbage that no power should induce me to lend myself to any of his quarrels … and that I should myself communicate in a direct manner with the editors … He was *furious*. I imperturbable … I only want you to understand that all my *time* and my *energy* have been miserably absorbed the last few days; for what between Babbage and the editors both pressing hard in different directions, I have been torn to pieces … [36]

Angry or not, Lovelace remained focused on the central issue that the specific purpose of the translation and "Notes" was to advance the actual building of the machine, rather than again to attack the government. In a candid letter to Babbage she offered her talents and resources to pursue the building of the Analytical Engine, provided he himself would stick to the technical aspects of the project. From Ockham Park on Monday 14 August 1843, Lovelace wrote to Babbage:

> I have now touched on all the grounds which can be taken on the supposition of its *really being pernicious to your interests* that I have thus allowed the article to appear … My moral standard, such as it is, I must stick to; as long as it *is* my moral standard. … I *have* a right to expect from you the belief that I do sincerely and honestly take this view. [I]f *your* knowledge of *me* does not furnish sufficient grounds for doing so, then I can only say that no natural knowledge of any two human beings in this life can give fixed and stable grounds for faith and confidence then Adieu to *all* truth and to everything most generous in this world!
>
> I must now come to a practical question respecting the future. …
>
> If I am able to lay before you in the course of a year or two explicit and honourable propositions for *executing your engine* (such as are approved by persons whom you may *now* name to be referred to for their approbation) would there be any chance of your allowing myself and such parties to conduct the business for you; your own *undivided* energies being devoted to the execution of the work; all these matters being arranged for you on terms which your *own* friends should approve?
>
> You will *wonder* over this last query. But I strongly advise you not to reject it as chimerical. You do *not* know the grounds I have for believing that such a contingency may come within my power and I wish to know before I allow my mind to employ its energies any further on the subject, that I shall not be wasting thought and power for no purpose or result … Yours ever most sincerely A.A.L. [37]

A letter she wrote to her mother the next day confirms that the printers were recognizing her as author of the "Notes." Tuesday, 15 August:

> … I was unexpectedly summoned by the printers who needed a further supervision and as it is *actually to be out* I understand tomorrow, there was no time for *post* communications. No one can estimate the trouble of *interminable* labour of having to revise the printing of *mathematical* formulae. You will receive a few copies (amongst a hundred that are printed separately for me). …
>
> If he [Babbage] does consent to what I propose, I shall probably be enabled to keep him out of much hot water; and to bring his engine to *consummation* (which all I have seen of him and his habits the last 3 months, makes me scarcely anticipate it ever *will* be, unless someone really exercises a strong co-ercive influence over him). He is beyond measure *careless* and *desultory* at times. … [38].

With the final material delivered to the printer and after months of 18-hour days spent describing the possibilities of an extraordinarily complex virtual machine, Lovelace now confessed herself often very tired. Lovelace came up to London around 18 August to meet Babbage. He was still furious about not having had his own way on the idea of appending a diatribe to the "Notes." He scribbled a curt memo in the margin of Lovelace's letter of 14 August: "Saw AAL this morning and refused all the conditions." Instead of using publication of the *Memoir* with the "Notes" as a descriptive model of a strategy for gaining public understanding and support to get the Engine financed and constructed, Babbage would continue until his death in 1871 to go his own, often irascible, way.

By 24 August 1843, the volume of *Taylor's Scientific Memoirs* with the translation of Menabrea's "Memoir" and the "Notes" appeared. Lovelace wrote to her mother: "We are by no means desirous of making it [Author's note:

authorship of the "Notes"] a secret although I do not wish the importance of the thing to be exaggerated and overrated." [39] Charles Wheatstone wrote on 25 August 1843:

> My Dear Lady Lovelace, I called yesterday at the printer's and was informed that a separate copy of your paper had been forwarded by post to Ockham, and the new number of the Scientific Memoirs sent to St. James' Square … Yours very truly C. Wheatstone [40]

Reaction to the work was swift and positive. The paper, so Michael Faraday, famous for his chemical and electrical experiments, declared to Babbage on 1 September, was so complex it was well over his own head [41]. Menabrea asked Babbage to pass along his congratulations "à cette noble Dame, A.A.L." [42] With congratulations pouring in, even Babbage was pleased, and he swiftly reconciled with Lovelace, concluding a letter to her of 12 September 1843, with the extravagant: "Ever my fair Interpretess Your faithful slave C. Babbage." [43]

Babbage expert, Doron Swade (having examined the extensive exchange of letters and the resulting "Notes"), when interviewed for *To Dream Tomorrow*, commented:

> Ada saw something that Babbage in some sense failed to see. In Babbage's world his engines were bound by number. He saw that the machines could do algebra in the narrow sense that they could manipulate plus and minus signs. But all his calculating engines, his Difference Engine and his Analytical Engine, which is the programmable general-purpose machine, were all bound by number. They manipulated number as a manifestation of quantity, as a measure of quantity. What Lovelace saw--what Ada Byron saw--was that number could represent entities other than quantity. So once you had a machine for manipulating numbers, if those numbers represented other things, letters, musical notes, then the machine could manipulate symbols of which number was one instance, according to rules. It is this fundamental transition from a machine which is a number cruncher to a machine for manipulating symbols according to rules that is the fundamental transition from calculation to computation--to general purpose computation--and looking back from the present high ground of modern computing, if we are looking and sifting history for that transition, then that transition was made explicitly by Ada in that 1843 paper.

As Swade is fully aware, "[T]he Analytical Engine," as A.A.L. so clearly stressed, "does not occupy common ground with mere 'calculating machines'." This formulation, based on what only existed as a virtual machine in 1843, went beyond any known statement of Babbage, and beyond distinguished predecessors in mechanical calculation such as Blaise Pascal and Gottfried Wilhelm Leibniz. A.A.L. anticipated advanced work in the next century of Alan Turing, Konrad Zuse, Howard Aiken, Grace Hopper, and John von Neumann. Looking far ahead to that time when a general-purpose machine would no longer be declared worthless but would in fact be built, Lovelace argued that such a machine would serve as a springboard for an ever-increasing number of discoveries, many of which would remain unimaginable until such time as the machine was built and could be run. She wrote in Note A:

> [V]ery valuable practical results would be developed by the extended powers of the Analytical Engine, some of which would be brought forth by the daily increasing requirements of science and by a more intimate practical acquaintance with the powers of the engine, were it in actual existence. [44]

Lovelace was to be proven right, but it would take over 100 years. Only after the early ENIAC ("a computer of the Babbage type," as H.J. Gray described it) was built to run rapid calculations for ballistics tables did engineers and programmers, such as John von Neumann, began to move beyond what Lovelace had called "mere calculating machines" and begin, in Swade's words, "to manipulate symbols according to rules." With these developments in the mid-20th century, the paradigm shift Lovelace had made in 1843 would start to become our everyday reality.

References and notes

[1] British Library, London, additional manuscript (hereafter "add'l ms.") 40, 514, folio 223.

[2] A. A. Lovelace, "Notes by A.A. L. [Augusta Ada Lovelace]," *Taylor's Scientific Memoirs*, London, vol. III, 1843, pp. 666-731. These notes are printed in both *Charles Babbage and His Calculating Engines: Selected Writings by Charles Babbage and Others*, P. Morrison and E. Morrison, eds., Dover Publications, 1961 (which includes the full text of the Menabrea translation and the 1843 Notes, pp. 225-297), and in *Faster Than Thought*, B.V. Bowden, ed., Sir Isaac Pitman & Sons, Ltd, 1953, pp. 341-408. A. A. Lovelace's translation of Menabrea together with her "Notes" are also on the Web: http://www.fourmilab.ch/babbage/sketch.html.

[3] For information about the film, see http://www.flarefilms.org

[4] P. Morrison and E. Morrison, eds., *Charles Babbage and His Calculating Engines* ..., p. 249.

[5] Ibid., p. 252.

[6] She is, of course using the customary plural of scholarly writers of the time.

[7] British Library, add'l ms. 37, 192 folios 189-194.

[8] British Library, add'l ms. 37, 192 folio 326.

[9] A. Hyman, *Charles Babbage, Pioneer of the Computer*, Princeton Univ. Press, 1982, p. 227.

[10] I. B. Cohen, *Howard Aiken: Portrait of a Computer Pioneer*, MIT Press, p.63

[11] Cited by D. Swade, *Charles Babbage and His Calculating Engines*, Science Museum, 1991, p. 34

[12] Cited from H.J. Gray's *Digital Computer Engineering*, Prentice-Hall, 1963, in the annotated bibliography given by B. Randell, ed., *The Origins of Digital Computers: Selected Papers*, Springer Verlag, 1973, p. 420

[13] A. Hodges, *Alan Turing: The Enigma*, Vintage, p. 304

[14] Ibid., p. 297 and pp. 357-358.

[15] Cited by D. Swade, *The Cogwheel Brain*, pp. 160-161.

[16] Byron/Lovelace Collection, Bodleian Library, Oxford, UK, box 168, folio 47, recto and verso.

[17] Cited by B.A. Toole, *Ada, the Enchantress of Numbers*, p. 225. We might note that as we did our own transcriptions of letters, in a number of cases our reading and dating differs from Toole's. Ideally, all these documents should be put directly on the Web so that various people can do their own decipherments of texts that are often extremely hard to read. In some cases, Toole has very usefully included facsimiles of some of the handwritten documents, an excellent practice.

[18] Byron/Lovelace Collection, Bodleian Library, box 168, folio 43, recto and verso.

[19] Byron/Lovelace Collection, Bodleian Library, box 168, folio 45, recto and verso, and folio 46, recto.

[20] British Library, add'l ms. 37, 192, folio 337.

[21] British Library, add'l ms. 37, 192, folio 344.

[22] British Library, add'l ms. 37, 192, folio 348.

[23] British Library, add'l ms. 37, 191, folio 638

[24] British Library, add'l ms. 37, 192, folios 351-352

[25] British Library, add'l ms. 37, 192, folio 354.

[26] British Library, add'l ms. 37, 192, folios 357-358.

[27] British Library, add'l ms. 37, 192, folio 362.

[28] British Library, add'l ms. 37, 192, folio 386-387

[29] See D.A. Stein, *Ada, A Life and a Legacy*, MIT Press, 1985, p. xi, where much is made of " *her* [emphasis added] curiously ignored translation of a printer's error." Stein claimed that if Lovelace missed a proofreading error, she must have been unsound in mathematics. Stein then argues we must see Babbage as the primary author of the "Notes" (though he, too missed the printer's error) and see Lovelace's centrality as what Stein calls a "mythology." However, from the surviving letters of Babbage, Lovelace, Wheatstone, Lyell, Faraday, Menabrea, and the editors of *Taylor's Scientific Memoirs*, it is clear that proofreading was done with Babbage, and that the original typesetting error was missed by Menabrea. The original documentation shows that all contemporaries of Lovelace and Babbage having first-hand knowledge of how the "Notes" came into being, acknowledged Lovelace as the primary author.

[30] British Library, add'l ms. 37, 192, folios 393-394.

[31] British Library, add'l ms. 37, 192, folios 398-399.

[32] British Library, add'l ms. 37, 192, folios 414-415.

[33] Letter is given in full in B.A. Toole, *Ada*, pp. 219-222.

[34] Byron/Lovelace Collection, Bodleian Library, box 168, folios 41 and 42.

[35] D. Swade, *Cogwheel*, p. 163.

[36] Byron/Lovelace Collection, Bodleian Library, box 42, folio 76.

[37] British Library, add'l ms. 37, 192, folios 425-426.

[38] Byron/Lovelace Collection, Bodleian Library, box 42, folios 86-88.

[39] Byron/Lovelace Collection, Bodleian Library, box 42, folios 101-102.

[40] British Library, add'l ms. 54, 089, folio 37.

[41] British Library, add'l ms. 37, 192, folio 445.

[42] British Library, add'l ms. 37, 192, folio 46.

[43] British Library, add'l ms. 54, 089, folio 54.

[44] P. Morrison and E. Morrison, eds., *Charles Babbage and His Calculating Engines* …, p. 256.

Bibliography

B.V. Bowden, ed. (1953), *Faster Than Thought*, Sir Isaac Pitman & Sons, Ltd. Includes the text of Lovelace's translation of Menabrea and the full text of her "Notes."

M. Campbell-Kelly and W. Aspray (1996), *Computer*, Basic Books.

I.B. Cohen (1999), *Howard Aiken: Portrait of a Computer Pioneer*, MIT Press.

J. Fuegi and J. Francis (2003), *To Dream Tomorrow*, documentary film with Doron Swade, Sadie Plant, Miranda Seymour, David Herbert, Michael Lindgren, and direct Lovelace desendent, the Earl of Lytton; http://www.flarefilms.org.

D. Herbert (1997), *Lady Byron and Earl Shilton*, Hinckley and District Museum, Leicestershire.

A. Hodges (1992), *Alan Turing: The Enigma*, Vintage.

A. Hyman (1982), *Charles Babbage, Pioneer of the Computer*, Princeton Univ. Press.

M. Lindgren (1987), *Glory and Failure: The Difference Engines of Johann Müller, Charles Babbage, and Georg and Edvard Scheutz*, translated from Swedish by C.G. McKay, Linköping.

A.A. Lovelace's Translation of Menabrea together with her "Notes" are on the web: http:///fourmilab.ch/babbage/sketch.html.

P. Morrison and E. Morrison, eds. (1961), *Charles Babbage and His Calculating Engines: Selected Writings by Charles Babbage and Others*, Dover Publications. Includes the full text of the Menabrea translation and the 1843 Notes.

S. Plant (1997), *zeros + ones: Digital Women + the New Technoculture*, Doubleday.

R. Randell, ed. (1973), *The Origins of Digital Computers*, Springer Verlag. Contains a number of vital historical papers and a lengthy, superbly annoted bibliography.

J. Shurkin (1984), *Engines of the Mind*, W.W. Norton & Co.

D. Swade (2000), *The Cogwheel Brain*, Little, Brown and Co. The US edition of the same book: *The Difference Engine: Charles Babbage and the Quest to Build the First Computer*, Viking, 2001.

B.A. Toole (1992), *Ada, The Enchantress of Numbers*, Strawberry Press. Has the best collection of Ada Byron Lovelace letters now in print.

About the authors

John Fuegi has held American Council of Learned Societies, Guggenheim and Rockefeller Awards and has taught at Harvard University and the Freie Universität, Berlin. He was the Clara and Robert Vambery Distinguished Professor of Comparative Studies at the University of Maryland, College Park and a Maryland Institute for Technology in the Humanities (MITH) Fellow at the time this article was first published. He founded and currently co-chairs Flare Productions.

Jo Francis taught for many years in Thailand and the US before joining Flare Productions, a not-for-profit educational film-making organization which she currently co-directs. She was a Networked Fellow of the Maryland Institute for Technology in the Humanities during its founding years, has written on the teaching of history, and has received national and international recognition for her teaching and her work in film.

Readers may contact John Fuegi and Jo Francis at jf@flarefilms.org

Authors' note, 2015:

We are pleased that this article, originally written shortly after we completed *To Dream Tomorrow*, is being published again during this 200th anniversary year of Lovelace's birth. We have taken the opportunity to make a few small changes and updates. *To Dream Tomorrow* has also been re-issued, with new music in a few places to replace excerpts that originally had time-limited rights, so the documentary can continue to be available for viewing and screening, long-term.

Source Code Analysis of Flight Software using a SonarQube based Code Quality Platform

Maurizio Martignano

Spazio IT – Soluzioni Informatiche, San Giorgio di Mantova, Italy.

Andraes Jung

European Space Agency, Noordwijk, The Netherlands.

Tobias Lehmann

Inopus, Unterhaching, Germany.

Christian Schmidt

AIRBUS Helicopters, Donauwörth, Germany.

Abstract

Since 2012, Spazio IT and Inopus have been working on a code quality platform for the analysis of both Ada and C/C++ flight software.

For Ada (e.g. AIRBUS Helicopters NH90 and Tiger flight software), the emphasis has been on maintenance and particularly on the adoption of ISO/IEC 25010:2011 software characteristics to identify critical areas in large Ada codebases.

For C/C++ (e.g. European Space Agency IXV on board software), the emphasis has been on verification and validation, standards/guidelines enforcement and bug finding. Of particular interest is the development of a methodology able to apply in an effective way model checking and abstract interpretation techniques to large C/C++ code bases.

This paper describes both activities and shows the central role that SonarQube and Spazio IT developed plugins have plaid in their execution.

Keywords: Static Analysis, Ada, Quality Model, Characteristic, Metric, Measure, Maintenance, Maintainability, C/C++, Bug Finding, Bounded Model Checking, Abstract Interpretation, CBMC, Frama-C.

1 A Quality Model for Ada Maintainability

1.1 Quality Models

Various quality models are currently in use in embedded, real time and avionics systems. Some of the most used are:

- ISO/IEC 9126-1:2001 – "Software engineering -- Product quality -- Part 1: Quality model"
- ISO/IEC 25010:2011 – "Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models"
- ECSS-Q-HB-80-04A:2011 – "Software metrication programme definition and implementation" – European Cooperation for Space Standardization – [3]
- SQUALE:2012 – Software QUALity Enhancement – [8]

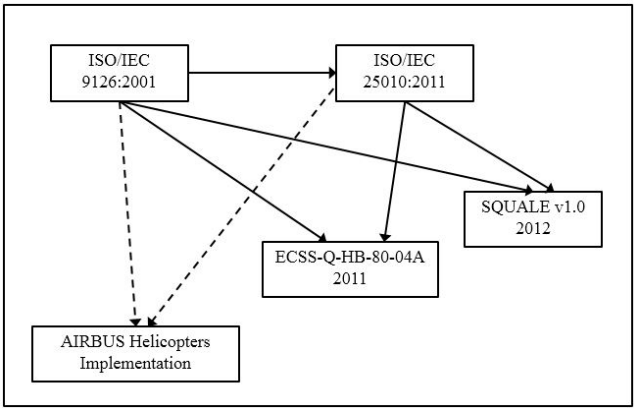

Figure 1- Quality Models

Figure 1 shows the above mentioned quality models and their relationships: an arrow from A to B means that B derives from A.

The ISO/IEC quality models are used in many application domains; ECSS-Q-HB-80-04A is mostly used in space applications and SQUALE usage is increasing everywhere (especially in systems written in Java, C/C++, C# and PHP) thanks to SonarQube (see the website [12]) success as code quality platform.

All models describe the quality of a software product/system as the result of a set of "characteristics", like functional suitability, reliability, maintainability… Characteristics may be defined in terms of "sub-characteristics": e.g. maintainability according to ISO/IEC 9126 consists of analysability, changeability, stability, testability, and so on… Characteristics and sub-

characteristics are eventually defined as functions of some measures, i.e. the values corresponding to pre-defined sets of metrics., e.g. the number of code lines, the number of comment lines, the number of subprograms in a package, etc,,,

Some of the metrics are related to source "code" entities e.g. the cyclomatic complexity, the nesting, etc... some others are related to other "non-code" software artefacts, e.g. the traceability between the system level requirements and the software level requirements/software design.

1.2 Maintainability Quality Model at AIRBUS Helicopters

The maintainability quality model defined by AIRBUS Helicopters and applied to the NH90 and Tiger flight software is a code-only (that is based on metrics only related to source code) quality model and mostly derives from ISO/IEC 9126 and ISO/IEC 25011 standards (see reference document [9]).

According to AIRBUS Helicopters quality model, the "maintainability index" is a function of four characteristics, i.e. analysability, changeability, stability and testability, and namely:

$$MI = g_A * A + g_C * C + g_S * S + g_T * T$$

with

$$g_A = g_{BA} * \frac{(C + S + T)}{3}$$

$$g_C = g_{BC} * \frac{(A + S + T)}{3}$$

$$g_S = g_{BS} * \frac{(C + A + T)}{3}$$

$$g_T = g_{BT} * \frac{(C + A + S)}{3}$$

The basic weights g_{BA}, g_{BC}, g_{BS} and g_{BT} are used to set the relative importance to the four characteristics; their sum equals one and currently they are all the same.

Each single characteristic is defined in turn as:

$$Characteristic = \frac{\sum_{n=1}^{m} MetricFullfillmentIndex_n}{m}$$

with

- $m =$ $number\ of\ metrics\ for\ one\ characteristic$
- $\text{MetricFullfillmentIndex}_n = \frac{\sum_{k=1}^{l} fullfilling(Metric_n, MetricScopeElement_k)}{l}$

where the metric fulfilment index of a given metric (e.g. "cyclomatic complexity") applied to a given element (e.g. a "subprogram" in the case of cyclomatic complexity) is the count of how many times its measure has an acceptable value divided by the total count of elements (e.g. the count of how many subprograms have cyclomatic complexity less than fifteen divided by the total count of subprograms).

These are the metrics used to compute the various characteristics (they are grouped by characteristic – their definitions are not repeated):

- Analysability
  - Count Lines – number of all lines
  - Nesting – maximum nesting level of control constructs
  - Count Declared Subprograms – number of declared subprograms
- Changeability
  - Cyclomatic Complexity – McCabe cyclomatic complexity
  - Nesting
  - Code Duplication – number of duplicated code lines
  - Count Declared Subprograms
  - Declarative Lines of Code – number of lines containing declarative source code
- Stability
  - Knots – measure of overlapping jumps
  - Executable Lines of Code - number of lines containing executable source code
  - Code Duplication
  - Cyclomatic Complexity
- Testability
  - Count Path – number of possible paths, not counting abnormal exits and "goto"s
  - Count Declared Subprograms
  - Comment to Code Ratio = ratio of comment lines to code lines.

Measures are collected by SCITOOLS Understand (see ref. website [11]); their aggregation into characteristics and in the final maintainability index is performed by the Spazio IT SonarQube Ada Plugin.

1.3 Spazio IT SonarQube Ada Metric Plugin

Spazio IT has developed for AIRBUS Helicopters a SonarQube Ada plugin in support of maintenance activities performed on large code bases (see ref. website [14]).

SonarQube is an open source web application that:

- takes in input a set of source code files and a set of analyses results (produced by external tools);

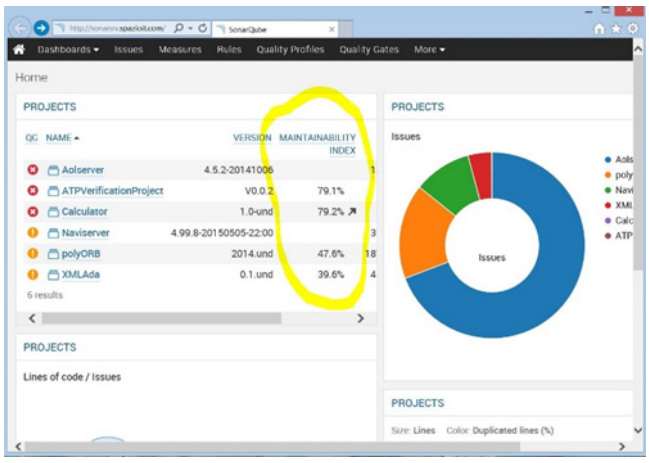

Figure 2 – Spazio IT SonarQube Ada Metric Plugin

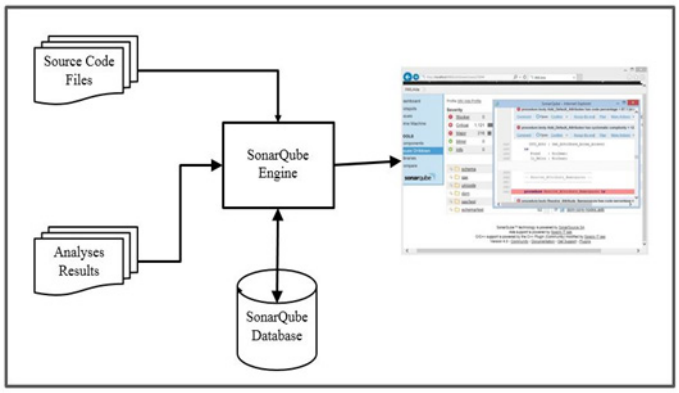

Figure 3 – SonarQube Architecture

- stores both sources and results in a database;
- makes available the gathered information via a dynamic website where the results are shown in the context of the code itself.

Analyses on the same code base can be performed at different moments in time and SonarQube keeps track of the changes/evolution. The problems found during analyses (a.k.a. issues) can be managed directly from within the system itself, e.g.

- identifying false positives
- assigning issues to developers;
- checking their status (if they have been solved).

1.4 SonarQubeAda Metric Plugin Results

The maintainability index has proved to correspond well with the «experienced» actual maintainability of real case projects.

Examples are available at this reference website [15] (see the open source AdaCore projects XMLAda and polyORB – of course, this demo server only contains open source examples and not actual flight software systems).

The issues found by the tool identify the elements, the points requiring a fix to improve the overall maintainability of the analysed projects. The plugin can also detect code duplication and display test and coverage data.

2 Independent Verification and Validation of the IXV on-board Software

2.1 Spazio IT C/C++ Code Quality Platform

"The Intermediate eXperimental Vehicle (IXV) is an European Space Agency (ESA) experimental re-entry vehicle to validate European reusable launchers. IXV successfully completed its 100-minute mission on the 11th of February 2015, being the first lifting body to perform full atmospheric re-entry from orbital speed. (see ref. doc [7])"

Spazio IT was requested to perform an activity of Independent Verification and Validation on the entire IXV on-board Software.

To this purpose Spazio IT integrated the open source code quality platform SonarQube with the following tools:

- CppCheck (see ref. website [2]) – open source – a C/C++ static analyser
- PC-Lint (see ref. website [10]) – proprietary - a rich pattern matching source code static analyzer (mostly used for MISRA C 2004 compliancy checks).

This integration was achieved by modifying the SonarQube C/C++ Community Plugin (see ref. website [13]).

Spazio IT also integrated the following more advanced and "research" tools to see if they were applicable to the IXV software and could provide additional information:

- CBMC (see ref. website [1]) – open source – a C prover based on bounded model checking
- Frama-C (see ref. website [4]) – open source – a framework for the static analysis of C code – especially its "value analysis" (i.e. abstract interpretation) and "weakest precondition calculus" plugins.

Apart from finding and removing issues in the flight software, Spazio IT has developed a methodology, which if effective in terms of bugs finding and allows for the application of CBMC and Frama-C to the analysis of large C/C++ code bases.

2.2 Developed Methodology

The developed methodology is divided in two parts:

- basic core – about how to use at best the compiler, CppCheck and PC-Lint
- model checking and abstract interpretation – about how to use at best CBMC and Frama-C.

2.2.1 Basic Core

- Identify which checks need to be executed on the code, i.e.
  - for the compiler, which compiler warnings (possibly all of them) need to be verified;
  - for CppCheck, which type of messages (errors, warnings, performance messages, and so on) need to be verified
  - for PC-Lint, which rule sets have to be used (e.g. MISRA C 2004), and for each rule set, which actual rules make sense and need to be verified
- Configure carefully the tools (in terms of tools options, selected memory model, location of the sources, location of the include files, and so on…
- Tune/optimize the configuration identified in the previous point by running few analysis sessions to verify that the proper information is generated (and disable the production of useless, noisy

outputs – this may require the development of some filtering scripts).

- Run the analyses whenever it makes sense in the lifetime of a project (or during operations), and possibly on a regular basis.
- At every analysis the code:
  - should compile;
  - should compile without generating any of the selected warnings;
  - should pass CppCheck analyses without generating any of the selected messages;
  - should pass PC-Lint analyses without violating any of the selected rules/guidelines.

2.2.2 Model Checking and Abstract Interpretation

CBMC and Frama-C Plugins (Value Analysis and Weakest Precondition) organize their computation into two phases:

- Generation of a model of the code under analysis
- "Symbolic execution" or "logic verification" of the model itself.

The computation resources required by phase one grow in a polynomial way with the complexity of code under analysis (number of files, packages, classes, functions, parameters, variables, lines of code, loops, constructs and so o…)

The computation resources required by phase two grow exponentially with the complexity of the code under of analysis.

So, for not so small, real code bases:

- either the analysis is stopped at the end of phase one
- or the system under analysis needs to be partitioned into reasonable, manageable "chunks".

CBMC phase one has shown to be good enough to prove the lack of infinite loops in the IXV code.

Using manageable "chunks", that is acting locally, at function/subprogram level has allowed both CBMC and Frama-C to detect issues in terms of:

- pointer checks;
- memory leak checks;
- signed/unsigned overflow;
- float overflow.

2.3 Found Issues

The following is a brief list of the types of issues found in the IXV source code. Each type of issue is accompanied by the tool that actually detected it.

- Uninitialized Variables
  - PC-Lint
- Array Index out of bounds
  - PC-Lint in all code bases but only in simple cases
  - CBMC and Frama-C in all possible cases but in small portions of code
- Constant Value Boolean Expression (MISRA C 2004 Rule 13.7)
  - PC-Lint
- Combining Signing and Unsigned Integers (MISRA C 2004 Rules 10.1, 10.3, 10.4)
  - PC-Lint
- Implicit integer type conversion (and promotion) (MISRA C 2004 10.1, 10.3, 10.4, 10.6, 10.7, 10.8)
  - PC-Lint
- Floating point comparison (MISRA C 2004 Rule 13.3)
  - PC-Lint
- Problems with pointers
  - PC-Lint
  - CBMC / Frama-C
- Divisions by Zero / Overflows
  - PC-Lint
  - CBMC / Frama-C
  - Traps

3 The Way Ahead

3.1 Quality Models and ALM Systems

All «software artefacts» can be represented as «collections of composite objects», i.e. objects containing hierarchies/trees of other objects… e.g.

- A requirements document contains requirements…
- A system consists of subsystems/modules, which in turn consist of packages, containing subprograms…

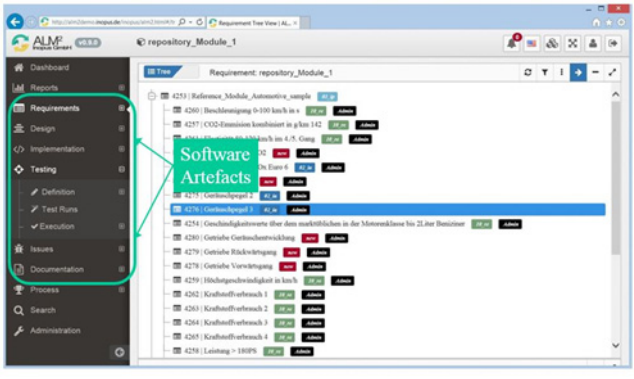

Figure 3 - Software Artefacts managed by Inopus ALM² system

- During a test campaign a set of tests are executed; these tests are described in test procedure documents and produce test data…

All «non-code» characteristics/metrics can be expressed as relationships among items/elements of these hierarchies/compositions… e.g.:

- A requirement is said «implemented» if there is a component in the software design (and eventually in the source code) actually implementing it.
- A requirement is said «verifiable» if there is a test able to prove that the requirement has been met.

Having all items/software artefacts stored and maintained in a single repository together with all specified relationships would be a clear advantage.

In fact, this repository would allow to know at any moment the actual (quality) status of the project.

Tools like IBM Doors or Inopus ALM^2 (see Inopus ALM^2 website [5] and demo server [6]) could be used as Requirements and/or Application Lifecycle Management System to create and maintain such single repository.

3.2 SonarQube: Bugs Finding and Knowledge Sharing

Code Quality Platforms like SonarQube have proved to be very valuable not only to endorse standards/guidelines but also and especially to:

- improve the efficiency of code inspection activities in finding and removing bugs;
- spread/share in an organization/corporate the culture, awareness, know-how related to a given programming language when used in a particular application domain.

The adoption of Code Quality Platforms should be encouraged in all software projects.

3.3 ALM Systems and SonarQube Integration

In the same way as in a quality model there are «code» and «non-code» characteristics/metrics, in order to manage in a complete and effective way the quality of a project it is necessary to combine a «code» quality platform together with a «non-code» quality platform.

Inopus and Spazio IT are currently working together to integrate ALM^2 («non-code» – see ref. demo server [6]) with SonarQube («code» – see ref. demo server [15]) and build a complete quality management system.

References

[1] CBMC, http://www.cprover.org/cbmc/.

[2] CppCheck, http://cppcheck.sourceforge.net/.

[3] European Cooperation for Space Standardization, http://ecss.nl/.

[4] Frama-C, http://frama-c.com/.

[5] Inopus ALM^2, http://www.inopus.de/ index.php?option=com_content&view=article&id=128&Itemid=574&lang=en

[6] Inopus ALM^2 Demo Server, http://alm2demo.inopus.de/login.html.

[7] IXV page at Wikipedia, http://en.wikipedia.org/wiki/Intermediate_eXperimental_Vehicle

[8] J.L. Letouzey, The SQUALE Definition Document, v1.0,http://www.sqale.org/wp-content/uploads/2010/08/SQALE-Method-EN-V1-0.pdf.

[9] C. Schmidt, B. Gumbel, *Software Measurement Regulations*, AIRBUS Helicopters Doc. Number 16SUZ00021E01.

[10] PC-Lint, http://www.gimpel.com/html/pcl.htm.

[11] SCITOOLS Understand, https://scitools.com/.

[12] SonarQube, http://www.sonarqube.org/.

[13] SonarQube C/C++ Community Plugin, https://github.com/wenns/sonar-cxx.

[14] Spazio IT Code Quality Platforms, http://www.spazioit.com/pages_en/sol_inf_en/code_quality_en/.

[15] Spazio IT SonarQube Demo Server, http://sonarsrv.spazioit.com/.

Support of Ravenscar in SPARK 2014

Steve Baird, Claire Dross, Yannick Moy, Tucker Taft
AdaCore

Florian Schanda
Altran UK

Abstract

This document presents the envisioned support for concurrent programming on a monoprocessor or a multiprocessor in the context of SPARK 2014, based on the existing Ravenscar profile of Ada.

The main goal of this support is to provide support for concurrent computation by expanding SPARK's supported subset of Ada to include tasks and protected objects (subject to some restrictions) while continuing to ensure statically the absence of run-time errors, data flow traceability, and the other benefits of SPARK.

This includes statically ensuring the absence of

- data races (i.e., unsynchronized access to shared data); and

- deadlocks; and

- run-time errors associated with the newly allowed language constructs (e.g., the run-time check associated with the Ravenscar profile's Max_Entry_Queue_Length restriction).

In most cases, this is accomplished by imposing rules which can be checked during flow analysis or earlier (as opposed to by introducing new verification conditions).

In the case of a monoprocessor application, the detection of deadlock depends on the Ceiling Protocol enforced by Ravenscar. In the case of a multiprocessor application, the detection of deadlock depends on a suitable Ceiling Protocol being followed, or on another mechanism like tasks following a fixed access order for protected objects.

In the future, SPARK may be further extended to support other concurrency profiles that are being discussed in the context of IRTAW (International Real-Time Ada Workshop), which lift some restrictions of Ravenscar that users have found problematic (e.g. maximum of one entry per protected object).

1 Proposed Tasking Model in SPARK

Tasks may communicate with each other via synchronized objects; these include protected objects, suspension objects, atomic objects, and "read-only after elaboration" objects (described later).

Other objects are said to be unsynchronized and may only be referenced (directly or via intermediate calls) by a single task (including the environment task) or by the protected operations of a single protected object.

SPARK's Part_Of aspect is generalized to support specifying this "ownership" relationship between unsynchronized global objects and their associated synchronized "owners". We call "virtual protected object" such an unsynchronized object (possibly volatile) whose access is mediated by a protected object. This allows ensuring the absence of data races without generating new verification conditions; the hazard is avoided solely through data flow analysis.

Similar techniques (although not implemented using the Part_Of aspect) are used to ensure that only a single task ever suspends on a given suspension object or calls an entry of a given protected object.

The same contract-related aspects are defined for a protected subprogram or entry as for an unprotected subprogram (an entry is treated like a procedure). The protected object itself is treated as an "in" parameter to protected functions, and an "in out" parameter to protected procedures and entries.

SPARK's Global and Depends aspects may also be specified for a task unit and have the same meaning that SPARK already defines for them in the case of a nonreturning subprogram. Thus, the rule preventing (for example) a task from accessing an unsynchronized global variable (unless the variable's Part_Of aspect indicates that such access is permitted) becomes a rule about the Global aspect of a task unit. Task nontermination is also ensured using the same rules that are already used for a nonreturning subprogram. Refined_Global and Refined_Depends aspects may be specified for a task body.

Similarly, SPARK's existing language rules for dealing with volatile objects are generalized to apply to synchronized objects. For example, a call to a protected function is subject to the same restrictions that sequential SPARK already imposes on reading a volatile object.

The notion of a volatile function is introduced (e.g., Ada.Real_Time.Clock or Ada.Task_Identification. Current_Task are volatile functions). A call to a volatile function is subject to the same restrictions that sequential SPARK already imposes on reading a volatile object. The implementation of a volatile function is allowed, for

example, to call a protected function or to read a volatile variable, and return statements for volatile functions are added to the "non-interfering context" list (more specifically, the expression of a return statement or the rhs of an assignment to the return object of an extended return statement are added to the list). A volatile function (unlike a non-volatile function) is allowed to have an effectively volatile object as a global input or as a formal parameter (although the Effective_Reads aspect must be False). A new Boolean aspect Volatile_Function is defined to identify such functions.

The notion of a synchronized state abstraction is introduced. The Boolean aspect used to specify this property of a state abstraction is named "Synchronous" because "Synchronized" is an Ada reserved word. A constituent (either an object or another state abstraction) of a given state abstraction shall be synchronized if and only if the state abstraction is synchronized.

Something functionally similar to Ada.Task_Attributes could be provided eventually in order to have a mechanism for accessing task-specific state. Note that the Ravenscar profile includes

```
No_Dependence => Ada.Task_Attributes,
```

so simply supporting Ada.Task_Attributes "as is" won't work. The Callable, Caller, Count, and Terminated attributes are supported, and modelled as reading the global external state Ada.Task_Identification. Tasking_State, hence can only appear where a read of a volatile variable would be allowed. The Identity, Priority, and Storage_Size attributes are supported but introduce no such dependency.

The Ada RM says:

> During a protected action, it is a bounded error to invoke an operation that is potentially blocking.

To statically prevent this bounded error from occurring, flow analysis will be able to determine whether any given subprogram is potentially blocking. [The ARG is in the process of defining a Boolean-valued Potentially_Blocking aspect to indicate (as part of a subprogram's specification) whether a subprogram is potentially blocking (see AI12-0064). At some time after that definition is finalized, the new aspect will probably be included in SPARK.]

Overriding restrictions are defined for the Potentially_Blocking and Volatile_Function aspects which are analogous to the existing rule for the Extensions_Visible aspect:

> A subprogram whose Extensions_Visible aspect is True shall not override an inherited primitive operation of a tagged type whose Extensions_Visible aspect is False. [The reverse is allowed.]

Static prevention of deadlock is guaranteed by flow analysis. More specifically, flow analysis detects cyclic dependencies involving locking associated with calls to protected functions and procedures. Note that the program may still block on calls to protected entries and suspension objects, which are not covered by the deadlock detection. The priority checks associated with the Ceiling_Locking locking policy are handled in proof.

Functions such as Calendar.Clock are marked as volatile and are specified as taking an external state abstraction as a global input. Preconditions are added as appropriate to subprograms provided by packages such as Ada.Calendar, Ada.Real_Time, Ada.Execution_Time, in order to avoid language defined runtime check failures (e.g., Ada.Execution_Time.Clock should not be passed a null Task_Id).

Delay statements are allowed (subject to Ravenscar's No_Relative_Delay restriction). Synchronized tagged types (including synchronized interface types) are allowed.

Ravenscar includes the No_Task_Hierarchy and No_Task_Termination restrictions, which eliminates the problems associated with using a Task_Identification.Task_Id value after the associated task has terminated or no longer exists. Thus, we don't need to impose any restrictions to avoid "dangling" Task_Id values in the context of Ravenscar. SPARK does statically prevent (via flow analysis) violations of the rule

> It is a bounded error to call the Current_Task function from an entry_body, or an interrupt handler, or finalization of a task attribute.

Ada's Attach_Handler aspect takes an expression of type Interrupts.Interrupt_Id, but package Interrupts declares an access-to-subprogram type (and access-to-subprogram types are not currently in SPARK). This is dealt with by marking SPARK_Mode On/Off in the Ada.Interrupts spec.

An unsynchronized object whose Part_Of aspect specifies that it "belongs" to a protected unit is treated for purposes of state abstraction and flow analysis similarly to a component of the protected type. For example, it is not (directly) a part of the visible or hidden state of the enclosing package. Similarly, an unsynchronized object whose Part_Of aspect specifies that it "belongs" to a task unit is treated as though it were declared immediately within the task body. Restrictions are imposed which ensure that if an unsynchronized object "belongs" to a task unit or a protected unit then exactly one object of that type is declared (this is trivially satisfied in the case of an anonymous type).

An object of a task or protected type is treated the same as any other object with respect to the Global, Depends, Refined_Global, Refined_Depends, and Refined_State aspects. A protected object is treated like a record object with respect to modification of its components; for example, a procedure which calls a protected procedure as follows

```
Some_Global_Protected_Object.
Set_Some_Component (To_Value => 123);
```

would probably include Some_Global_Protected_Object on its list of In_Out globals. A task object is treated like a record object with respect to reading its discriminants (if any) and task-specific attributes. In contrast, local variables declared within a task body are not considered to be components of the task object. It would never make sense to list a top-level object of a task type as an In_Out global of a subprogram because a task object cannot be modified.

SPARK's anti-aliasing rules could be relaxed for some synchronized objects. These rules are not needed for avoiding data races (this follows from the definition of "synchronized object") and they are not needed for proofs. Roughly speaking, if a variables's value might spontaneously change at any time, then it doesn't matter for purposes of proofs if it also happens to be updated as a result of aliasing. It's not clear how much benefit might be gained by taking advantage of this.

2 Proposed Language Restrictions

Language restrictions beyond those imposed by the Ravenscar profile include:

1. Synchronized objects (as defined above) may only be declared at library level. [Ravenscar requires this in some but not all cases.]
2. Variables referenced (directly or through intermediate subprogram call) by two or more tasks or protected objects shall be synchronized.
3. A function cannot (directly or through intermediate subprogram calls) suspend or delay.
4. Either all or none of the components of an object shall be synchronized. [Corner case: an extension of a componentless tagged type shall not have a synchronized component.]
5. A Partition_Elaboration_Policy of Sequential is required. In addition to preventing premature task activation, this is also needed in order to allow tasks to safely access "read-only after elaboration" objects. These are variables which are modified only during library unit elaboration and can be viewed as constants after task activation has begun. A new Boolean aspect Constant_After_Elaboration is defined to identify such objects.
6. No synchronized ghost objects.
7. A protected type shall define full default initialization. A "virtual protected object" (i.e., an object whose Part_Of aspect indicates that it can only be accessed via the protected operations of one protected object) must similarly either be imported, have an explicit initial value or be of a type which defines full default initialization.

3 Proposed Modifications to the Standard Library

3.1 Ada.Execution_Time

1. Package spec is marked SPARK_Mode => On (private part is SPARK_Mode => Off).
2. Function Clock is marked Volatile_Function with a global input of Ada.Task_Identification. Tasking_State.
3. A precondition is added to Clock: Task_Id /= Task_Identification.Null_Task_Id.
4. A precondition is added to functions "+" and "-" on CPU_Time to ensure that the result fits in the result type.
5. A precondition is added to Time_Of to ensure that the result should fit in the result type.
6. Function Clock_For_Interrupts is marked Volatile_Function with a global input of Ada.Task_Identification.Tasking_State.
7. A precondition is added to Clock_For_Interrupts: Interrupt_Clocks_Supported = True.

3.2 Ada.Execution_Time.Interrupts

1. A precondition is added to Clock: Separate_Interrupt_Clocks_Supported = True
2. A postcondition is added to Clock: (if not Supported (Interrupt) then Clock'Result = Ada.Execution_Time.Time_Of(0))

3.3 Ada.Interrupts

1. Functions Is_Reserved, Is_Attached, and Get_CPU are marked SPARK_Mode => On (note that the package spec cannot be marked SPARK_Mode On as Parameterless_Handler is defined as an access type) and other subprograms are marked SPARK_Mode => Off.
2. Functions Is_Attached and Get_CPU are marked Volatile_Function with a global input of Ada.Task_Identification.Tasking_State.
3. A global input-output of Ada.Task_Identification. Tasking_State is added to procedure Detach_Handler.

3.4 Ada.Real_Time

1. Package spec is marked SPARK_Mode => On (private part is SPARK_Mode => Off)
2. An external abstract state Clock_Time is added to package Ada.Real_Time
3. Function Clock is marked Volatile_Function with a global input of Clock_Time.
4. A precondition is added to arithmetic operators on Time and Time_Span to ensure that the result fits in the result type.
5. A preconditions is added to To_Duration, To_Time_Span, Nanoseconds, Microseconds,

Milliseconds, Seconds, and Minutes to ensure that the result fits in the result type.

6. A precondition is added to Time_Of to ensure that the result should fit in the result type.

3.5 Ada.Real_Time.Timing_Events

This entire package (which fundamentally depends on a visible access-to-subprogram type) is not in SPARK. This is a stronger restriction than Ravenscar's No_Local_Timing_Events restriction.

3.6 Ada.Synchronous_Task_Control

1. Package spec is marked SPARK_Mode => On (private part is SPARK_Mode => Off)
2. All procedures have a dependency S => null (despite in out mode for S)
3. Function Current_State is marked Volatile_Function with a global input of Ada.Task_Identification. Tasking_State.
4. Procedure Suspend_Until_true is marked Potentially_Blocking.

3.7 Ada.Task_Identification

1. It defines an external abstract state Tasking_State with Async_Readers => True and Async_Writers => True (but Effective_Reads => False and Effective_Writes => False). This state is used to model access to the runtime system by various standard functions (for example Current_Task in that same unit) and attribute references T'Identity and E'Caller.
2. Package spec is marked SPARK_Mode => On (private part is SPARK_Mode => Off)
3. Function Current_Task is marked Volatile_Function with a Global Input of Ada.Task_Identification.Tasking_State.
4. Procedure Abort_Task is marked SPARK_Mode Off (note that Ravenscar forbids calling Task_Identification.Abort_Task with restriction No_Abort_Statements).

4 Impact on Legality Checking

The frontend is the part of GNATprove which is shared with the GNAT compiler. SPARK legality rules are enforced in the frontend, for those parts of a program that are marked SPARK_Mode => On.

In addition to enforcing Ravenscar restrictions when the Ravenscar profile is set, the frontend and the part of GNATprove checking SPARK legality rules will enforce some of the basic rules that do not require full flow analysis:

- Restrictions on calling context of volatile functions (we already do this for volatile objects).
- Enforcement of library-level declarations for synchronized objects.
- The correct partition elaboration policy is set.
- Enforcing the none/all-components-are-synchronized rule.
- Rejecting synchronized ghosts.
- Demanding full default initialization for protected types (including its virtual state).

Additionally, Ravenscar profile should be set whenever a concurrency construct is in a part of code marked SPARK_Mode => On.

5 Impact on Flow Analysis

While SPARK 2005 required that all information related to tasks and protected objects appear in package specs to make modular flow analysis possible, SPARK 2014 does not make this simplification and thus flow analysis for tasking will be non-modular initially. However, we do expect to add contracts for all tasking related issues so that it is possible to return to a fully modular analysis; since some of the contracts are currently also discussed by the ARG it seemed like a good idea to wait until we have a standard set. Note that unlike computation of globals, the analysis required is much simpler (simple graph connection problems). The following properties will be computed:

- Suspension objects suspended-on (for single-suspender restrictions).
- Protected entries called (for single-caller restrictions).
- Unsynchronized objects read/written (for race conditions).
- Protected objects read-locked (for deadlock).
- Protected objects write-locked (for deadlock).
- Protected type instantiations (for singleton protected object restrictions).
- Task type instantiations (for singleton task object restrictions).
- Subprograms called directly or indirectly that access a protected object and the protected object accessed (for verification of the ceiling protocol).
- Which subprograms are potentially blocking (for absence of blocking in protected operations).

These will be used by the majority of checks that flow analysis will perform. Checks will be performed at three levels: during subprogram analysis (unit), during package analysis (package), and when analyzing overall graph (global). Errors computed during the global phase will be issued when we process the enclosing offending object.

The checks performed by flow analysis will be:

- (unit) No blocking in protected operations - a simple test if any called subprogram is one that might block. The error will be issued at the called subprogram, and we can probably point at which subprogram actually makes the call blocking (this might be quite far down the call tree).
- (global) When two or more tasks access any object, making sure that it is synchronized (each object will have 'owners', we will then test for this when the object is declared). The error will be raised when we analyze the enclosing subprogram or package.
- (unit|package) While the frontend checks that Part_Of correctly identifies the 'owner' of an unsynchronized object, flow analysis will make sure that any such objects are only used in their owner. Combined with the above, the check that no two tasks use the same unsynchronized object emerges.
- (global) Flow analysis will add a check that only a single task suspends on any given suspension object. The error will reference the package and suspension object. It probably makes sense to issue the error on the suspension object and provide the set of tasks suspending on it in the message.
- (global) Flow analysis will add a check that only a single task calls a specific entry of any given protected object. Error message as above.

Flow analysis will also check the new contract indicating an object may not be modified after elaboration, this check is similar to the existing check of making sure an `in' global is never modified.

Finally, CFG construction in flow analysis will need to understand the new syntax introduced by tasking: protected objects will be treated as always-private (potentially discriminated) records and tasks will be treated the same (although variables within a task T are not considered part of T when T appears in annotations).

6 Impact on Proof

Proof will be enhanced to support verification of the newly allowed language constructs by modelling adequately possible concurrent accesses and by generating new Verification Conditions where needed to complete the checks performed by flow analysis.

6.1 Modelling Concurrent Accesses

Proof of a task unit should be similar to the proof of a non-returning procedure. Namely, Verification Conditions will be generated for checking absence of run-time errors and non-termination (the latter is checked by simulating an assertion of False at the end of a task body, which should never be reachable).

Proof of a protected unit should be similar to the proof of a package. Verification Conditions will be generated for each of the unit's subprograms and entries. Proof of entry bodies will be treated like procedures.

Since they can be accessed and modified asynchronously during the execution of a subprogram, entry or task body, synchronized objects should be treated like volatile variable with Async_Writers => True and Async_Readers => True (but Effective_Writes => False and Effective_Reads => False). In other words, proof cannot assume that a synchronized object keeps its value between two successive accesses to read or write it.

The only exception to the above rule is for access to a protected object when proving one of its protected subprogram or entry. In that case, the protected object should be treated as a normal object not subject to concurrent access.

6.2 Generation of New Verification Conditions

New Verification Conditions will be generated for a few tasking specific run-time errors:

1. Checking that the expression of a pragma Attach_Handler is never reserved is done in proof as it requires dealing with values of expressions.
2. More noticeably, verifications of values of priorities will also be done in proof. For a protected object with either an Attach_Handler or an Interrupt_Handler aspect specified for one of its procedures, a Verification Condition will be generated to make sure that the ceiling priority of the object is in System.Interrupt_Priority.
3. Verification Conditions to check the Ceiling Protocol are generated during the verification of tasks bodies. At each call to a subprogram which accesses, directly or indirectly, a protected object, a Verification Condition will be generated to check that the active priority of the thread is less or equal to the ceiling priority of the object.

7 Examples

The following example is the stopwatch example from RavenSPARK translated into SPARK 2014.

```
-- tuningdata.ads
with System, Ada.Real_Time;

package TuningData with SPARK_Mode
is
   -- priorities
   UserPriority : constant
      System.Interrupt_Priority := 31;
   TimerPriority : constant System.Priority := 15;
   DisplayPriority : constant
      System.Interrupt_Priority := 31;

   -- task periodicities
   TimerPeriod : constant Ada.Real_Time.Time_Span
      :=  Ada.Real_Time.Milliseconds (1000);
```

```
end TuningData;

-- display.ads
with TuningData;

package Display with SPARK_Mode,
  Abstract_State => (State with External =>
      (Async_Readers, Effective_Writes))
is
   procedure Initialize with
     Global  => (In_Out => State),
     Depends => (State => State);

   procedure AddSecond with
     Global  => (Output => State),
     Depends => (State => null);

end Display;

-- display.adb
with System.Storage_Elements;

package body Display with
  SPARK_Mode,
  Refined_State => (State => Internal_State)
is
   -- External variable Port is a virtual protected
   -- object. All accesses to Port are mediated by
   -- protected object Internal_State, which is
   -- specified with the Part_Of aspect on Port.
   Port : Integer with
     Volatile,
     Async_Readers,
     Effective_Writes,
     Address =>
      System.Storage_Elements.To_Address
                (16#FFFF_FFFF#),
     Part_Of => Internal_State;

   protected Internal_State with
     Interrupt_Priority => TuningData.DisplayPriority
   is
      -- add 1 second to stored time and send it to port
      procedure Increment with
        Global  => null,
        Depends => (Internal_State => Internal_State);

      -- clear time to 0 and send it to port;
      procedure Reset with
        Global  => null,
        Depends => (Internal_State => null);

   private
      Counter : Natural := 0;
   end Internal_State;

   protected body Internal_State is
      procedure Increment is
      begin
         Counter := Counter + 1;
         Port := Counter;
      end Increment;

      procedure Reset is
      begin
         Counter := 0;
         Port := Counter;
      end Reset;
   end Internal_State;

   procedure Initialize with
     Refined_Global  => (In_Out => Internal_State),
     Refined_Depends => (Internal_State =>
       Internal_State)
   is
   begin
      Internal_State.Reset;
   end Initialize;

   procedure AddSecond with
     Refined_Global  => (Output => Internal_State),
     Refined_Depends => (Internal_State => null)
   is
   begin
      Internal_State.Increment;
   end AddSecond;

end Display;

-- timer.ads
with TuningData;
limited with Ada.Synchronous_Task_Control,
     Ada.Real_Time, Display;

package Timer with
  SPARK_Mode,
  Abstract_State => (Oper_State, Timing_State)
is

   -- These two procedures simply toggle
   -- suspension object Operate
   procedure StartClock with
     Global  => (Output => Oper_State),
     Depends => (Oper_State => null);

   procedure StopClock with
     Global  => (Output => Oper_State),
     Depends => (Oper_State => null);

end Timer;

-- timer.adb
with Ada.Synchronous_Task_Control,
     Ada.Real_Time, Display;
use type Ada.Real_Time.Time;

package body Timer with
  SPARK_Mode,
  Refined_State => (Oper_State => Operate,
     Timing_State => TimingLoop)
```

```
   is
      Operate :
Ada.Synchronous_Task_Control.Suspension_Object;

      task TimingLoop with
        Global  => (Output => Oper_State,
                    In_Out => Display.State,
                    Input  => Ada.Real_Time.Clock_Time),
        Depends => (Oper_State    => null,
                    Display.State =>+ null,
                    null          => Ada.Real_Time.Clock_Time),
        Priority => TuningData.TimerPriority;

      task body TimingLoop is
        Release_Time : Ada.Real_Time.Time;
        Period : constant Ada.Real_Time.Time_Span :=
          TuningData.TimerPeriod;
      begin
        Display.Initialize; -- ensure we get 0 on the screen
                            -- at start up
        loop
          -- wait until user allows clock to run
          -- calling procedure Suspend_Until_True
          -- which is Potentially_Blocking
          Ada.Synchronous_Task_Control.
                Suspend_Until_True (Operate);
          Ada.Synchronous_Task_Control.
                Set_True (Operate);
          -- once running, count the seconds
          -- calling Ada.Real_Time.Clock which is a
          -- Volatile_Function
          Release_Time := Ada.Real_Time.Clock;
          Release_Time := Release_Time + Period;
          delay until Release_Time;
          -- each time round, update the display
          Display.AddSecond;
        end loop;
      end TimingLoop;

      procedure StartClock
      is
      begin
        Ada.Synchronous_Task_Control.
          Set_True (Operate);
      end StartClock;

      procedure StopClock
      is
      begin
        Ada.Synchronous_Task_Control.
          Set_False (Operate);
      end StopClock;

   end Timer;

   -- user.ads
   with TuningData;
   limited with Timer, Display;

   package User with
     SPARK_Mode,
     Abstract_State => Button_State
   is
   end User;

   -- user.adb
   with Timer, Display;

   package body User with
     SPARK_Mode,
     Refined_State => (Button_State => Buttons)
   is
     protected Buttons is
       pragma Interrupt_Priority
        (TuningData.UserPriority);

       procedure StartClock with
         Global  => (Output => Timer.Oper_State),
         Depends => (Timer.Oper_State => null),
         Attach_Handler => 1;

       procedure StopClock with
         Global  => (Output => Timer.Oper_State),
         Depends => (Timer.Oper_State => null),
         Attach_Handler => 2;

       procedure ResetClock with
         Global  => (In_Out => Display.State),
         Depends => (Display.State =>+ null),
         Attach_Handler => 3;
     end Buttons;

     protected body Buttons is
       procedure StartClock
       is
       begin
         Timer.StartClock;
       end StartClock;

       procedure StopClock
       is
       begin
         Timer.StopClock;
       end StopClock;

       procedure ResetClock
       is
       begin
         Display.Initialize;
       end ResetClock;
     end Buttons;
   end User;

   -- main.adb
   with User, Timer, Display, Ada.Real_Time;

   procedure Main with
     SPARK_Mode,
     Global  => (Input  => Ada.Real_Time.Clock_Time,
                 In_Out => (User.Button_State,
```

```
                Timer.Oper_State,
                Display.State)),
  Depends => (User.Button_State =>+ null,
              Timer.Oper_State =>+ User.Button_State,
              Display.State    =>+ (Timer.Oper_State,
                  User.Button_State),
              null             => Ada.Real_Time.Clock_Time),
                  Priority => 10
is
begin
  null;
end Main;
```

References

[16] A. Burns, B. Dobbing and T. Vardanega (2004), *Guide for the use of the Ada Ravenscar Profile in high integrity systems*, University of York Technical Report.

[17] SPARK (2010), *The SPARK Ravenscar Profile*.

[18] T. Taft, B. Moore, L. M. Pinho and S. Michell, *Safe Parallel Programming in Ada with Language Extension*, Proc. of the 2014 ACM SIGAda annual conference on High integrity language technology, pp. 87-96, ACM.

[19] A. Burns and A. J. Wellings (2013), *Locking Policies for Multiprocessor Ada*, ACM SigAda Letters v. 33 issue 2, pp. 59-65, ACM.

[20] S. Lin (2013), *A Flexible Multiprocessor Resource Sharing Framework for Ada*, PhD thesis, University of York.

SPARK 2014 Rationale: Ghost Code, Object Oriented Programming and Functional Update

Yannick Moy

AdaCore, France

Abstract

This paper continues the publication of the "SPARK 2014 Rationale", which started in the December 2013 issue of the Ada User Journal. In this instalment, we present three contributions regarding ghost code, Object Oriented programming and functional update in SPARK.

1 Ghost Code

A common situation when proving properties about a program is that you end up writing additional code whose only purpose is to help proving the original program. This ghost code may be code that expresses the very properties you want to prove, or code that allows naming in properties some quantities that would have no name otherwise. This is more common when proving richer properties (for example an integrity, functional or security property), but this may also be needed when proving a "mundane" property like absence of run-time errors.

If you're careful or lucky enough, the additional code you write will

- not impact the program being verified, and
- be removed during compilation, so that it does not inflate binary size or waste execution cycles.

But SPARK provides a better way, by marking the corresponding code as ghost code, using the new Ghost aspect. This instructs GNATprove to check property 1 above and GNAT to ensure property 2 above. For example, a function that is only used in contracts and assertion pragmas can be marked as ghost as follows:

```
function Is_Valid (X : T) return Boolean with Ghost;
```

and a variable that is only used to store or compute values read in contracts and assertion pragmas can be marked as ghost as follows:

```
Current_State : State_T with Ghost;
```

Besides declarations of ghost entities (packages, subprograms, types or variables), ghost code consists also of:

- statements that assign to a ghost variable,
- calls to ghost procedures,
- contracts and assertion pragmas that refer to a ghost entity.

The identification of ghost code in SPARK is thus mostly syntactic, which ensures both that ghost code is visibly ghost to programmers and easily removed by GNAT during compilation. At the same time, SPARK has verification rules that ensure that ghost code cannot impact the functional behavior of non-ghost code, and that users cannot partly disable updates to a ghost variable (either all updates must be enabled, or they must be disabled). GNATprove checks those rules.

On the one hand, ghost code in SPARK is mostly targeted at facilitating proofs, a direct descendant from auxiliary variables used in the 60s (see the Related Work section of the article The Spirit of Ghost Code [1]). On the other hand, ghost code in Ada or SPARK provides a strong mechanism for safe code instrumentation, reminiscent of aspect programming principles.

For more examples of uses of ghost code, see the section of SPARK User's Guide on Ghost Code [2].

For an advanced use of ghost code to perform manual proof, see [3].

2 Object Oriented Programming

Object Oriented Programming is known for making it particularly difficult to analyze programs, because the subprograms called are not always known statically: this is the well-known feature of dispatching calls, a.k.a. late binding, whereby the target of the call depends on the dynamic type of the object on which it's called. For example, the standard for civil avionics certification has recognized this specific problem, and defines a specific verification objective called Local Type Consistency that should be met with one of three strategies (DO-332 document, paragraph OO.6.7.2):

4. Verify substitutability using formal methods.
5. Ensure that each class passes all the tests of all its parent types which the class can replace.
6. For each call point, test every method that can be invoked at that call point (pessimistic testing).

SPARK allows using strategy 1 above, by defining the behavior of an overridden subprogram using a class-wide contract (introduced by aspects Pre'Class and Post'Class) and checking that the behavior of the overriding subprogram (also defined using a class-wide contract) is a suitable substitution. What is a suitable substitution? One that satisfies the Liskov Substitution Principle (a.k.a. LSP, named after Barbara Liskov who defined it in an article

with Jeannette Wing in 1993), which essentially says that the behaviors of the overriding subprogram are a subset of the possible behaviors of the overridden subprogram. When programming by contract is used, as in SPARK, this translates as two essential properties:

1. The precondition of the overriding subprogram should be equal or less restrictive than the precondition of the overridden subprogram.
2. The postcondition of the overriding subprogram should be equal or give more guarantees than the postcondition of the overridden subprogram.

For example, assume that you have a tagged type Object that defines a procedure Draw to display the object on screen. The Draw procedure may require that the object is fully visible, and may set a flag in the object to record that it has been drawn:

```
type Object is tagged record ...
procedure Draw (Obj : in out Object) with
  Pre'Class  => Is_Included (Obj, Screen),
  Post'Class => Is_Drawn (Obj);
```

Then, Object may be derived in a number of specific objects, say a box, which define their own Draw procedure. In order to respect LSP, these new procedures should have preconditions that can only weaken the precondition on Object, and postconditions that can only strengthen the postcondition on Object. For example, here is an overriding of Draw that respects LSP:

```
type Box is new Object with record ...
procedure Draw (Obj : in out Box) with
  Pre'Class  => not Is_Empty (
     Intersect (Obj, Screen)),
  Post'Class => Is_Drawn (Obj) and
     Canonical_Form (Obj);
```

Assuming that a box always occupies some space, if it is included in the screen then its intersection with the screen is not empty, hence the precondition of the overriding subprogram is indeed weaker than the precondition of the overridden one. For the postcondition, the one of the overriding subprogram is the same as the one of the overridden subprogram, plus an additional property, hence it is stronger.

An overriding of Draw that does not respect LSP could either fail to keep or weaken its precondition:

```
procedure Draw (Obj : in out Box) with
  Pre'Class  => Is_Centered (Obj, Screen),  -- BAD
  Post'Class => ...
```

or fail to keep or strengthen its postcondition:

```
procedure Draw (Obj : in out Box) with
  Pre'Class  => ...
  Post'Class => Canonical_Form (Obj);  -- BAD
```

GNATprove can be used to check automatically that overriding subprograms respect LSP, and will detect the bad overridings above. The benefit of enforcing LSP is that the same class-wide contract can be used to analyze calls to all possible targets of a dispatching call. Thus, proof of code with dispatching calls is no more complex than proof of code with regular calls, except the class-wide contract is used for dispatching calls instead of regular contracts for regular calls.

For more details on how Object Oriented programs can be verified with GNATprove, see the SPARK User's Guide [2].

3 Functional Update

While attribute Old allows expressing inside postconditions the value of objects at subprogram entry, this is in general not enough to conveniently express how record and array objects are modified by a procedure. A special attribute Update is defined in SPARK to make it easy to express such properties.

As mentioned in a previous post [4], attribute Old allows expressing inside a postcondition the value of an object at subprogram entry. For example, the postcondition of the procedure Incr can be written:

```
procedure Incr (X : in out Integer) with
  Post => X = X'Old + 1;
```

This is fine for a scalar variable, but what about a composite variable? If X is a record with 3 integer components A, B and C, we may write:

```
procedure Incr (X : in out Rec) with
  Post => X.A = X.A'Old + 1;
```

and if X is an array of integers, we may write:

```
procedure Incr (X : in out Arr) with
  Post => X(1) = X(1)'Old + 1;
```

This is fine for specifying the value of the record component or array element which has been incremented, but what about others? As humans, we may read implicitly in the contracts above that components other than A, and elements other than at index 1, have not been modified by calling Incr. But the analysis tool GNATprove cannot rely on that implicit information, as the same contracts may be correct for procedures that do modify components B and C and elements at indexes different from 1. Hence, GNATprove interprets the contracts above as:

> *X is an "in out" parameter that can be modified by calling Incr, and the only thing we know about that is how the value of component A (or the element at index 1) is modified. We don't know anything about how other components (or elements) are modified.*

The solution is to express explicitly in the postcondition the property that other components or elements are not modified by Incr:

```
procedure Incr (X : in out Rec) with
  Post => X.A = X.A'Old + 1 and then
          X.B = X.B'Old  and then
          X.C = X.C'Old;
```

```
procedure Incr (X : in out Arr) with
  Post => X(1) = X(1)'Old + 1 and then
        (for all J in X'Range =>
              (if J /= 1 then X(J) = X'Old(J)));
```

With these postconditions, GNATprove can use the fact that only component A and the element at index 1 are modified by calling Incr, when analyzing Incr's callers.

But the above postconditions are not so easy to read, and scale poorly if there are many more components, or if the modification is applied to a deeply nested component. This is why SPARK defines a special attribute Update which copies the value of a composite object (record or array) with some modifications. The above postconditions can be expressed equivalently:

```
procedure Incr (X : in out Rec) with
  Post => X = X'Old'Update (A => X.A'Old + 1);

procedure Incr (X : in out Arr) with
  Post => X = X'Old'Update (1 => X(1)'Old + 1);
```

This attribute is equivalent to the square bracket notation used in Ada 2005 for the same purpose. It can also be used with benefits to express functions whose only purpose is to return a slightly modified version of their input. For example (using the syntax of expression functions [2]):

```
function Incr (X : Rec) return Rec is
      (X'Update (A => X.A + 1));

function Incr (X : Arr) return Arr is
      (X'Update (1 => X(1) + 1));
```

Such usage is quite common in functional languages. For example both OCaml and Haskell have a special syntax for functional record update.

To know more about the attribute Update, see SPARK User's Guide [2] and SPARK Reference Manual [5].

References

[1] J-C. Filliatre, L. Gondelman and A. Paskevich (2014), *The Spirit of Ghost Code*, Proc. of the 26th Intl. Conf. on Computer Aided Verification, Vienna, Austria.

[2] AdaCore and Altran UK Ltd (2013), *SPARK 2014 Toolset User's Guide*.

[3] C. Dross (2014), *Manual Proof with Ghost Code in SPARK 2014*, http://www.spark-2014.org/entries/detail/manual-proof-in-spark-2014.

[4] Y. Moy (2013), *SPARK 2004 Rationale: Pre-call and Pre-loop Values*, Ada User Journal vol. 34, issue 4. http://www.spark-2014.org/entries/detail/spark-2014-rationale-pre-call-and-pre-loop-values.

[5] AdaCore and Altran UK Ltd (2013), *SPARK 2014 Reference Manual.*

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o KU Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Hubert B. Keller
Karlsruher Institut für Technologie (KIT)
Institut für Angewandte Informatik (IAI)
Campus Nord, Gebäude 445, Raum 243
Postfach 3640
76021 Karlsruhe
Germany
Email: Hubert.Keller@kit.edu
URL: ada-deutschland.de

Ada-France

attn: J-P Rosen
115, avenue du Maine
75014 Paris
France
URL: www.ada-france.org

Ada-Spain

attn. Sergio Sáez
DISCA-ETSINF-Edificio 1G
Universitat Politècnica de València
Camino de Vera s/n
E46022 Valencia
Spain
Phone: +34-963-877-007, Ext. 75741
Email: ssaez@disca.upv.es
URL: www.adaspain.org

Ada in Sweden

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

c/o Ahlan Marriott
Altweg 5
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: president@ada-switzerland.ch
URL: www.ada-switzerland.ch