ADA USER JOURNAL

Volume 35
Number 4
December 2014

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.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

This last issue of 2014 publishes the Proceedings of the "Workshop on Challenges and New Approaches for Dependable and Cyber-Physical System Engineering", which took place June 23, co-located with Ada-Europe 2014. This workshop brought together industry and research participants, for a full-day discussion on dependability and critical issues of Cyber-Physical Systems (CPS), a good complement to the already rich program of the conference.

The workshop program included 2 technical sessions with papers from academia and industry, an invited speech by Charles Robinson, of Thales, France, a focused session, and a roundtable discussion. The proceedings reflect part of this rich content, starting with an Editorial by Daniela Cancila and Jean-Louis Gerstenmayer, from CEA LIST, France, followed by a set of technical papers. The first workshop paper, from a group of authors from the University of York and Rapita Systems, UK, which discusses the use of controlled vocabulary and structured expressions for CPS in the automotive domain, to improve understanding between the different teams involved in the development process. The next paper, by authors from Krono-Safe, France, presenting Kron-OS, a real-time kernel, and the associated set of tools, which targets the development of safe mixed-criticality applications. Afterwards, authors from the National Institute of Informatics and DENSO Corporation, Japan, which presents a formal model of energy consumption behavior in mobile platforms, which can be used form contract-based analysis method to detect and remove energy-related bugs. Finally, the last workshop paper from authors from CEA LIST, Technological Research Institute SystemX – Alstom Transport, France, and the University of Trento, Italy, presents a feasibility study feasibility study on the use of contract-based approaches for enforcing safety-related properties in CPS. The proceedings close with a report on the round-table discussion that took place at the workshop.

The issue also continues the publication of the contents of the industrial track of Ada-Europe 2014, with a paper by Robert Cholay, describing the AdDoc tool, which was built both to generate documentation and also to check conformance to commenting rules, and that also provides a good example of the use of ASIS.

Finally, and as usual, the issue provides the News Digest, Calendar and Forthcoming Events sections, provided by the News and Events Editors, respectively Jacob Sparre Andersen and Dirk Craeynest. A special mention to the forthcoming events section, with information about the Ada Developer Room at FOSDEM 2015, 31 January 2015, Brussels, Belgium (I take the opportunity to congratulate Ada-Belgium for the important work on promoting Ada within the open source community), the always important International Real-Time Ada Workshop (IRTAW 2015) which will be held in Vermont, USA, April 2015; and obviously Ada-Europe 2015, which will take place at the Universidad Politécnica de Madrid, Spain, 22-26 June 2015: the deadline for submissions is around the corner.

Luís Miguel Pinho
Porto
December 2014
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 Organisations

Please Submit Contract Assertion Tests to ACATS

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 13 May 2014 16:26:37 -0500
Subject: Re: Dynamic_Predicate failure -> Assertion_Error?
Newsgroups: comp.lang.ada

[...]

Well, as this is a weird compiler bug and not an outright mistake in the implementation (since it depends solely on the bounds of the loop - it works properly when no loop is involved), it's not a good candidate for the ACATS. Especially as it seems to be more likely an exception processing problem rather than an assertion problem.

Moreover, AI12-0054-2 and AI12-0071-1 extensively changed the rules in this area (they were much too loose for practical usability). There will need to be tests for those AIs, but they have to wait until AI12-0071-1 is approved by WG 9 (expected in June).

That said, we'd love to have more tests for Ada 2012's contract assertions. A variety of programming styles helps the quality of the ACATS. Contact me at agent@ada-auth.org if you need more information, or see Annex E in the ACATS documentation (http://www.ada-auth.org/acats-files/3.1/docs/UG-E.HTM).

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]

Ada-Europe 2015

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Wed, 2 Jul 2014 21:41:26 +0000
Subject: CfP 20th Conf. Reliable Software Technologies, Ada-Europe 2015
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

Preliminary Call for Papers

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

22-26 June 2015, Madrid, Spain

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

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

*** CfP in HTML/PDF on web site ***

Ada-Europe organizes annual international conferences since the early 80's. This is the 20th event in the Reliable Software Technologies series, previous ones being held at Montreux, Switzerland ('96), London, UK ('97), Uppsala, Sweden ('98), Santander, Spain ('99), Potsdam, Germany ('00), Leuven, Belgium ('01), Vienna, Austria ('02), Toulouse, France ('03), Palma de Mallorca, Spain ('04), York, UK ('05), Porto, Portugal ('06), Geneva, Switzerland ('07), Venice, Italy ('08), Brest, France ('09), Valencia, Spain ('10), Edinburgh, UK ('11), Stockholm, Sweden ('12), Berlin, Germany ('13), and Paris, France ('14).

General Information

The 20th International Conference on Reliable Software Technologies - Ada-Europe 2015 will take place in Madrid, Spain. 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

11 January 2015: Submission of regular papers, tutorial and workshop proposals

25 January 2015: Submission of industrial presentation proposals

1 March 2015: Notification of acceptance to all authors

29 March 2015: Camera-ready version of regular papers required

12 April 2015: Industrial presentation abstracts required

17 May 2015: Tutorial and workshop materials 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.

Topics of interest to this edition of the conference include but are not limited to:

- Multicore and Manycore Programming: Predictable Programming Approaches for Multicore and Manycore Systems, Parallel Programming Models, Scheduling Analysis Techniques.
- Real-Time and Embedded Systems: Real-Time Scheduling, Design Methods and Techniques, Architecture Modelling, HW/SW Co-Design, Reliability and Performance Analysis.
- Mixed-Criticality Systems: Scheduling methods, Mixed-Criticality Architectures, Design Methods, Analysis Methods.
- Theory and Practice of High-Integrity Systems: Medium to Large-Scale Distribution, Fault Tolerance, Security, Reliability, Trust and Safety, Languages Vulnerabilities.
- Software Architectures: Design Patterns, Frameworks, Architecture-Centred Development, Component-based Design and Development.
- Methods and Techniques for Software Development and Maintenance: Requirements Engineering, Model-driven Architecture and Engineering,

Formal Methods, Re-engineering and Reverse Engineering, Reuse, Software Management Issues, Compilers, Libraries, Support Tools.

- Software Quality: Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems.
- Mainstream and Emerging Applications: Manufacturing, Robotics, Avionics, Space, Health Care, Transportation, Cloud Environments, Smart Energy systems, Serious Games, etc.
- Experience Reports in Reliable System Development: Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics.
- Experiences with Ada and its Future: Reviews of the Ada 2012 new language features, implementation and use issues, positioning in the market and in the software engineering curriculum, lessons learned on Ada Education and Training Activities with bearing on any of the conference topics.

Call for Regular Papers

Authors of regular papers which are to undergo peer review for acceptance are invited to submit original contributions. Paper submissions shall not exceed 14 LNCS-style pages in length. Authors shall submit their work via EasyChair following the relevant link on the conference web site. 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 start of the conference. The authors of accepted regular papers shall prepare camera-ready submissions in full conformance with the LNCS style, not exceeding 14 pages and strictly by March 29, 2015. 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 CiteSeerX Venue Impact Factor has the Conference in the top quarter. Microsoft Academic Search has it in the top third for conferences on programming languages by number of citations in the last 10 years. 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 January 25, 2015. Submissions shall be made via EasyChair following the relevant link on the conference web site. 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 April 12, 2015, 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 Chair directly.

Call for Tutorials

Tutorials should address subjects that fall within the scope of the conference and may be proposed as either half- or full-day events. 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 will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

Workshops on themes that fall within the conference scope may be proposed. Proposals may be submitted for half- or full-day events, to be scheduled at either end of the conference week. Workshop proposals should be submitted to the Conference Chair. The workshop organizer shall also commit to preparing proceedings for timely publication in the Ada User Journal.

Call for Exhibitors

The commercial exhibition will span the three days of the main conference. Vendors and providers of software products and services should contact the Exhibition Chair for information and for allowing suitable planning of the exhibition space and time

Grants for Reduced Student Fees

A limited number of sponsored grants for reduced fees is expected to be available for students who would like to attend the conference or tutorials. Contact the Conference Chair for details.

Organizing Committee

See CFP in Forthcoming Events section (pg. 243).

Enabling Safety Certification in ARM-based Systems

URL: https://event.on24.com/eventRegistration/EventLobbyServlet?target=registration.jsp&eventid=846366&sessionid=1&key=818DF79D54BFD1A3F14610A7B20BF1C8&partnerref=adacore

Enabling Safety Certification in ARM-based Systems

October 8, 2014 2:00 PM EDT

Processor technology from ARM has become a game changer for multiple industries, delivering high-performance-per-watt processing and high levels of integration to enable system on a chip (SoC) capability in a low-power device. This combination has been ideal for small form factor systems in avionics, automotive, and medical applications. Now embedded designers in these markets are looking at ways to take advantage of ARM technology to enable safety certification via standards such as FAA DO-178C for avionics systems and MISRA for automotive systems. This webcast of industry experts will look at how ARM-based solutions can not only reduce power but easily utilize the integrated peripherals in safety certification solutions across different industries.

Sponsors:

AdaCore, DDC-I

Moderator:

John McHale, OpenSystems Media

Linux Day 2014 in Cagliari

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Fri, 24 Oct 2014 17:55:07 +0200
Subject: Ada 2012 talk in Cagliari tomorrow
Newsgroups: comp.lang.ada

The Linux user group in Cagliari (GULCh) has invited me to give a talk on contract-based programming at the "Linux Day" conference tomorrow (http://linuxday.gulch.it/2014/).

I have promised to make the talk accessible to anybody with programming experience, but the examples and practical possibilities I will discuss are all based on

Ada 2012 (with a single SPARK 2014 exception ;-).

Everybody are welcome!

From: Martyn Pike <usenet@embeddedconsultinguk.com>
Date: Sun, 26 Oct 2014 11:31:12 +0000
Subject: Re: Ada 2012 talk in Cagliari tomorrow
Newsgroups: comp.lang.ada

> [...]

How many people attended this talk that you gave ?

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Sun, 26 Oct 2014 18:21:54 +0100
Subject: Re: Ada 2012 talk in Cagliari tomorrow
Newsgroups: comp.lang.ada

> [...]

I think it was somewhere between 30 and 40 people. [Confirmed by the organisers. —sparre]

I've definitely had a more crowded auditorium for an Ada talk in Cagliari, but that was ten years ago, and a possibly more attractive subject (GUI programming).

FOSDEM 2015

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Sun, 2 Nov 2014 11:13:21 +0000
Subject: CfP - Ada Developer Room at FOSDEM 2015, Brussels, Belgium
Newsgroups: comp.lang.ada, fr.comp.lang.ada

Call for Presentations

6th Ada Developer Room
at FOSDEM 2015

Saturday 31 January 2015,
Brussels, Belgium

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

Organized in cooperation with
Ada-Europe

Ada-Belgium [1] is pleased to announce that there will be a one-day Ada Developer Room on Saturday 31 January 2015 at FOSDEM 2015 in Brussels, Belgium. This Ada DevRoom is once more organized in cooperation with Ada-Europe [2].

General Information

FOSDEM [3], the Free and Open source Software Developers' European Meeting, is a free and non-commercial two-day weekend event organized early each year in Brussels, Belgium. It is highly developer-oriented and brings together 5000+ participants from all over the world. No registration is necessary. The goal is to provide open source developers and communities a place to meet with other developers and projects, to be informed about the latest developments in the open source world, to attend interesting talks and presentations on various topics by open source project leaders and committers, and to promote the development and the benefits of open source solutions.

Ada Developer Room

At previous FOSDEM events, Ada-Belgium has organized very well attended Ada Developer Rooms, offering a full day program in 2006 [4], a two-day program in 2009 [5], and full day programs in 2012 [6], 2013 [7] and 2014 [8]. An important goal is to present exciting Ada technology and projects also to people outside the traditional Ada community.

Our proposal for another dedicated Ada DevRoom was accepted, and now work continues to prepare the detailed program. We most probably will have a total of 8 schedulable hours between 10:00 and 18:00 in a room which holds some 60 participants. More information will be posted on the dedicated web-page on the Ada-Belgium site [9], and final announcements will of course also be sent to various forums, lists and newsgroups.

Call for Presentations

Ada-Belgium calls on you to:

- inform us at ada-belgium-board@cs.kuleuven.be about specific presentations you would like to see in this Ada DevRoom;
- for bonus points, subscribe to the Ada-FOSDEM mailing list [9] to discuss and help organize the details;
- for more bonus points, be a speaker: the Ada-FOSDEM mailing list is the place to be!

Do you have a talk you want to give?

Do you have a project you would like to present?

Would you like to get more people involved with your project? We're inviting proposals that are related to Ada software development, and include a technical oriented discussion. You're not limited to slide presentations, of course. Be creative. Propose something fun to share with people so they might feel some of your enthusiasm for Ada!

Speaking slots are 25 or 50 minutes, including Q&A. Depending on interest, we might also have a session with lightning presentations (e.g. 5 minutes each).

We'd like to put together a draft schedule early December. So, please act ASAP, and definitely by November 30, 2014 at the latest.

We look forward to lots of feedback and proposals!

Dirk Craeynest, FOSDEM Team of Ada-Belgium

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

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

[2] http://www.ada-europe.org

[3] https://fosdem.org

[4] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/06/060226-fosdem.html

[5] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/09/090207-fosdem.html

[6] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/12/120204-fosdem.html

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

[8] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/14/140201-fosdem.html

[9] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

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

Ada-related Resources

Repositories of Open Source Software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Mon Nov 3 2014
Subject: Repositories of Open Source software

AdaForge: 8 repositories [1]

Bitbucket: 109 repositories [2]
16 developers [2]

Codelabs: 20+ repositories [3]

GitHub: 654 repositories [4]
126 developers [5]

Rosetta Code: 606 examples [6]
28 developers [7]

Sourceforge: 241 repositories [8]

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

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

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

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

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

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

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

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

[See also “Repositories of Open Source Software”, AUJ 35-3, p. 153. —sparre]

Ada on Social Media

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Wed Nov 5 2014
Subject: Ada on Social Media

Ada groups on various social media:

- LinkedIn[1]: 2_052 members
- Reddit[2]: 726 readers
- Google+[3]: 348 members
- StackOverflow[4]: 264 followers
- Twitter[5]:1 twitter

[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 "Social Media Sites", AUJ 34-2, p. 64. —sparre]

Open Source Build Server Status

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Thu Nov 6 2014
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 "Experimental Continuous Integration System for Open Source Projects", AUJ 35-1, p. 6. —sparre]

Ada-related Tools

Statistics Libraries

From: Poul-Erik Andreasen <poulerik69@gmail.com>
Date: Mon, 07 Apr 2014 16:34:27 +0200
Subject: Statistics
Newsgroups: comp.lang.ada

What do people here use when they need statistics. I am specially interested in Probability Kernel Density functions.

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue, 8 Apr 2014 14:34:55 -0700
Subject: Re: Statistics
Newsgroups: comp.lang.ada

> [...]

You may want to have a look at MathPaqs:

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

there is a "samples" packages in the stats subdirectory. No KDE so far, though, just plain histograms. There are also some random simulation tools.

[See also "Excel Writer, GNAVI, Mathpaqs and Zip-Ada", AUJ 34-4, p. 200. —sparre]

From: Poul-Erik Andreasen <poulerik69@gmail.com>
Date: Wed, 09 Apr 2014 02:24:54 +0200
Subject: Re: Statistics
Newsgroups: comp.lang.ada

> [...]

That may be just what I need. I have decided to make the KDE myself. The math is not that awful. The formulas are on Wikipedia and most of what I need is in Ada.Numerics. I will take look at Mathpags to see if there is some useful stuff for me there.

From: Simon Wright <simon@pushface.org>
Date: Wed, 09 Apr 2014 07:37:30 +0100
Subject: Re: Statistics
Newsgroups: comp.lang.ada

> [...]

If you need asymmetric matrices, you might find Ada 2005 Math Extensions useful.

http://sourceforge.net/projects/gnat-math-extn/

[See also "Ada 2005 Math Extensions", AUJ 34-3, p. 138. —sparre]

From: Poul-Erik Andreasen <poulerik69@gmail.com>
Date: Wed, 09 Apr 2014 23:58:32 +0200
Subject: Re: Statistics
Newsgroups: comp.lang.ada

> [...]

I will take a look at it. The vector types may be useful.

[See also "Mathematics and Statistics", AUJ 34-4, p. 203. —sparre]

PDF Writer

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 10 Apr 2014 22:33:00 +0200
Subject: Re: Writing PDF files
Newsgroups: comp.lang.ada

> [...]

Gtk supports PDF surfaces in Cairo. For example:

http://www.dmitry-kazakov.de/ada/aicwl.htm#12.6

does plotting, in particular, into PDF.

In general, whatever output generated by Cairo (Cairo is vector graphics library used in Gtk), it can be rendered on a PDF surface, i.e. in a PDF file.

From: Bill Findlay <yaldnif.w@blueyonder.co.uk>
Date: Thu, 10 Apr 2014 22:47:05 +0100
Subject: Re: Writing PDF files
Newsgroups: comp.lang.ada

> [...]

I gave up on PDF and implemented a very small subset of Encapsulated PostScript that was good enough for my very simple requirements (emulating a Calcomp plotter of the early 19060s).

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Sun, 13 Apr 2014 01:46:59 -0700
Subject: Re: Writing PDF files
Newsgroups: comp.lang.ada

> soon a PDF with "hello world"...

http://sf.net/p/apdf/code/HEAD/tree/

More soon (or not soon)...

LZMA

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Mon, 30 Jun 2014 07:43:21 -0700
Subject: Re: Q: LZMA in Ada ?
Newsgroups: comp.lang.ada

Just to answer my own question ;-) : there are now *two* implementations for decoding LZMA. In chronological order, in the following libraries:

- Matreshka: http://forge.ada-ru.org/matreshka/browser/trunk/design/filters

  [See also "Matreshka", AUJ 35-1, p. 8. —sparre]

- Zip-Ada: http://unzip-ada.sf.net

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

STM32F4 Discovery

From: Roy Emmerich <roy.emmerich@gmail.com>
Date: Tue, 26 Aug 2014 15:38:33 -0700
Subject: STM32F4 Discovery, communication and libraries
Newsgroups: comp.lang.ada

I discovered Ada 2 days ago, so stick with me.

I am starting a business which will focus on creating a cheap, modular, open source data logger/controller usable across multiple domains. At the moment I am in the prototyping stage, using the following hardware:

1. STM32F4 Discovery board

2. MikroElektronika STM32F4 Discovery shield (http://www.mikroe.com/stm32/stm32f4-discovery-shield/)

3. Various MikroElektronika click boards (http://www.mikroe.com/click/):

 * GPS click board (ublox LEA-6S receiver)

 * microSD

 * RS485

 * RS232

 * Ethernet

I don't have experience in C/C++ but I do have a lot of experience in Java, python, structured text (read PLCs) and a few other bits and pieces. I REALLY don't want to develop in C. From what I can make out it looks like a nightmare once the code reaches any substantial size, which mine will. I've started quite a few beginner C books and never got very far before throwing in the towel. However what I've read about Ada has certainly caused me to sit up!

So far I have investigated the following high level language alternatives:

1. www.espruino.com (JavaScript)

2. www.micropython.org

3. www.eluaproject.net

At the moment I am forging ahead with Espruino because:

1. it is quick to get code on the processor as it is interpreted

2. interfacing with external hardware via SPI/I2C/UART is easy...except when you want to access on-chip functionality that isn't yet supported by the Espruino interpreter (which is aimed at STM32F1 powered Espruino board, partially ported to the STM32F4).

3. www.npmjs.org has so many libraries and examples of how to get things done (e.g. MODBUS RTU library...done) which translates to many willing hands/minds.

but I see dragons on the horizon. Here are a few:

1. It is not hard real-time

2. Although you can minify the code, I am uncertain whether everything will fit on when the code base grows.

3. JavaScript on microcontrollers has no track record.

In short, nice for tinkering/prototyping but probably not a wise choice for the long run.

Today I started chatting to Mike Silva over at EmbeddedRelated:

http://www.embeddedrelated.com/showarticle/617.php

For Ada to be a viable option for my project, this is what I think I need [with Mike's comments]:

1. [IN PROGRESS] Easy communication: SPI, I2C, Serial, Ethernet,

 [Mike] I know that AdaCore is working on comms libraries for the ARM Cortex M parts, but I don't know anything about the projected availability.

 [Roy] If they want adopters then they'd better get a move on!

2. [UNSOLVED?] Libraries/examples: MODBUS RTU/Eth at the very least

 I have yet to find a repository of libraries covering the major protocols (e.g. MODBUS, CAN, one-wire). There are quite a few in C. Would it be viable to just wrap these in Ada? It seems like a great short term solution but if we are using Ada to make things more stable, it hardly makes sense to use it merely to wrap (flakey) C libraries ;)

 [Mike] It is also true that you can link to C code in Ada with either thin or thick wrappers. A thin wrapper just converts each C function to an equivalent Ada subprogram, while a thick wrapper adds one or more higher-level layers on top of the basic subprograms.

3. [SOLVED] Direct access to chip functionality: STM32F4 RTC, Precision Time Protocol capabilities on chip, etc. I read that binding in C code is fairly easy? That would allow me to directly call the STM32 C drivers provided by STM?

 [Mike] In any case, you will have no problem accessing the chip hardware in Ada.

4. [SOLVED] Someone hosts an open forum to encourage the exchange of ideas, providing an alternative to the normally clandestine military/large corporate approach to code development. If Ada is going to grow then it needs to open up to your average Joe like me.

 [Mike] comp.lang.ada!

I'd appreciate any further feedback from members of this list.

From: Luke A. Guest <laguest@archeia.com>
Date: Wed, 27 Aug 2014 02:40:48 +0000
Subject: Re: STM32F4 Discovery, communication and libraries
Newsgroups: comp.lang.ada

You might be interested in <https://github.com/rowsail/AdaForMicrocontrollers> which is being discussed between some of us on LinkedIn in the Ada for micro controllers group.

From: Mike Silva <embeddedrelatedmike@scriptoriumdesigns.com>
Date: Wed, 27 Aug 2014 09:17:24 -0700
Subject: Re: STM32F4 Discovery, communication and libraries
Newsgroups: comp.lang.ada

The problem with reuse is that it is hardly ever as clean and simple as one would hope. In my experience, by the time you find some code, determine if it meets your needs, identify the areas that will need to be changed, and figure out how to bind to it if in C, it would have been quicker to write the code from scratch (perhaps using the code you found as a general guide). I don't claim that's a universal, just my experience.

For example, on the MODBUS drivers, I've used such code in the past, and even though our company paid for custom drivers, we spent a lot of time fixing and adjusting them. I doubt we gained anything over writing from scratch (using whatever code we could have found as a guide).

Speaking of reuse, since you're just discovering Ada, you should read about the Ariane 5 reuse fiasco (which some people foolishly tried to blame on the use of Ada, but which is really about the perils of reuse of perfectly good code).

From: Jonathan <johnscpg@googlemail.com>
Date: Fri, 29 Aug 2014 12:36:09 -0700
Subject: Re: STM32F4 Discovery, communication and libraries
Newsgroups: comp.lang.ada

[GNATColl contains Ravenscar support packages.]

The code itself looks intimidating, but most of the .ads files have sample code that shows how to use them. Here's the list:

gnatcoll-ravenscar-utils.ads

gnatcoll-ravenscar-utils.adb

gnatcoll-ravenscar-timers-one_shot_timer.ads

gnatcoll-ravenscar-timers-one_shot_timer.adb

gnatcoll-ravenscar-timers.ads

gnatcoll-ravenscar-timed_out_sporadic_server.ads

gnatcoll-ravenscar-timed_out_sporadic_server.adb

gnatcoll-ravenscar-sporadic_server_with_callback.ads

gnatcoll-ravenscar-sporadic_server_with_callback.adb

gnatcoll-ravenscar-sporadic_server.ads

gnatcoll-ravenscar-sporadic_server.adb

gnatcoll-ravenscar-simple_sporadic_task.ads

gnatcoll-ravenscar-simple_sporadic_task.adb

gnatcoll-ravenscar-simple_cyclic_task.ads

gnatcoll-ravenscar-simple_cyclic_task.adb

gnatcoll-ravenscar-multiple_queue_sporadic_server.ads gnatcoll-ravenscar-multiple_queue_sporadic_server.adb

gnatcoll-ravenscar-multiple_queue_cyclic_server.ads

gnatcoll-ravenscar-multiple_queue_cyclic_server.adb

gnatcoll-ravenscar.ads

Also <http://www.adacore.com/adaanswers/gems/gem-89-code-archetypes-for-real-time-programming-part-1/> might help.

Deepend

From: Brad Moore <brad.moore@shaw.ca>
Date: Sun, 07 Sep 2014 19:27:26 -0600
Subject: ANN: Deepend 3.4 Storage Pools
Newsgroups: comp.lang.ada

I am pleased to announce the availability of Deepend version 3.4.

Deepend is a suite of dynamic storage pools with subpool capabilities for Ada 95, Ada 2005, and Ada 2012. Bounded and unbounded storage pools types are provided. Storage pools with subpool capabilities allow all objects in a subpool to be reclaimed all at once, instead of requiring each object to be individually reclaimed one at a time. Deepend storage pools provides a more efficient and safer scheme for storage management than relying on the standard storage pool, and user calls to Unchecked_Deallocation. In fact, Deepend can eliminate the need for Unchecked_Deallocations. A Dynamic Pool may have any number of subpools.

Deepend can be downloaded from;

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

Differences since last release include;

This is technically the first version of Deepend that compiles for Ada 2012 and the GNAT GPL 2014 version of the compiler. In particular,

- The Pool parameter of the System.Storage_Pools.Subpools.Default_Subpool_For_Pool function was finalized to be an in out parameter for the Ada 2012 standard. This requires changes to the Deepend pools, since they override this function. In addition, the Ada 2005 and Ada 95 versions of Deepend also were modified to reflect this change. In Ada 95 and Ada 2005, functions cannot have in out parameters, so instead, the parameters were changed to be access parameters, so that the Ada 95 and Ada 2005 version more closely matches the Ada 2012 version.
- In the Ada 2012 version, there were static_predicates defined for private declarations, which in fact needed to be dynamic_predicates. Since these were private declarations, the predicates were removed, since they weren't very useful since they were private declarations, and the need for dynamic checks for this was deemed as worthwhile.
- Removed workarounds for GNAT compiler bugs that were fixed in the GNAT GPL 2014 version of the compiler. In particular, the storage pools have default discriminants which now can be left unspecified to use the defaults.

[See also “Deepend”, AUJ 35-1, p. 7. —sparre]

GNAT for More ARM Variants

From: gnlnops@gmail.com
Date: Mon, 8 Sep 2014 14:25:29 -0700
Subject: Re: GNAT SPARK:Embedded ARM Ada Project doesn't run in STM32F429 Discovery Board
Newsgroups: comp.lang.ada

If you are interested I perform the port of Ada runtime library and the demo_leds example. As Jerry wrote the RCC module is a little bit different on the STM32F42x and the origin of the problem came from the voltage scaling operation during initialization.

The main modifications were:

- PLL configuration,
- Add of the 9 new interrupt sources,
- Link command files,
- USART1 configuration update (from GPIOB to GPIOA).

For recall the pins are:

- LED3: PG13,
- LED4: PG14,
- USART1_TX: PA9,
- USART2_RX: PA10.

The LEDs and user button work correctly but I do not test the USART1 yet because I have no TTL<->RS232 converter.

The files are available on GitHub:

https://github.com/gnlnops/gnat-stm32f429i-disco

From: Brian Drummond <brian@shapes.demon.co.uk>
Date: Sat, 11 Oct 2014 10:53:49 GMT
Subject: Re: Newcomers to comp.lang.ada: welcome and how did you end up here ?
Newsgroups: comp.lang.ada

[...]

One piece of possibly good news: There's a LinkedIn thread where - just possibly - a critical mass of developers are getting together. Including Luke and others with some serious interest and past track record.

On this group, see the threads "Group development and porting of the RTS using GNAT GPL for ARM" and "http://www.AdaForMicrocontrollers.com now "Live".

https://www.linkedin.com/groups?home=&gid=2188035&trk=anet_ug_hm

They reference a currently not-very-lively forum:

https://www.adaformicrocontrollers.com/

and a Github repo:

https://github.com/rowsail/AdaForMicrocontrollers

[See also “Blog Entries on STM32F4 Programming”, AUJ 35-3, p. 162. —sparre]

Simple Components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 16 Sep 2014 22:08:21 +0200
Subject: ANN: Simple Components for Ada v4.2
Newsgroups: comp.lang.ada

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, multiple connections server designing tools. It grew out of needs and does not pretend to be universal. Tables management and strings editing are described in separate documents see Tables and Strings edit. The library is kept conform to the Ada 95, Ada 2005, Ada 2012 language standards.

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

Changes to the previous version:

- Transactional block files provided by the package Persistent.Blocking_Files.Transactional;
- Persistent.Memory_Pool provides task-safe access to the underlying container file;
- Persistent.Memory_Pool.Generic_External_B_Tree is changed to support multiple trees on the same pool;
- Various bug fixes and code cleanup.

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

Persistent Memory Pools

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 17 Sep 2014 18:51:21 +0200
Subject: Ada vs SQLite3 benchmark
Newsgroups: comp.lang.ada

I posted benchmark of Ada persistent B-tree vs. SQLite3 at Ada Programming blog:

http://ada-programming.blogspot.de/2014/09

The implementation of B-tree is based on Ada.Direct_IO with a transaction layer, e.g. for safety against system failure.

From: Emmanuel Briot <briot.emmanuel@gmail.com>
Date: Thu, 18 Sep 2014 01:08:46 -0700
Subject: Re: Ada vs SQLite3 benchmark
Newsgroups: comp.lang.ada

Why did you run the benchmarks without optimization? That seems inconsistent. If you are measuring performance, you should run with full optimization on I think.

Also, it would be interesting to use the following pragmas (combined or not) in SQLite, since they can impact performance significantly:

pragma journal_mode=WAL;

pragma synchronous=OFF;

(unless the Ada code is also running fsync() regularly)

I think the latter in particular will significantly change the time measured for SQLite.

But I agree with your conclusion that Ada is a viable alternative here, thanks for the experiment!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 18 Sep 2014 22:08:50 +0200
Subject: Re: Ada vs SQLite3 benchmark
Newsgroups: comp.lang.ada

> Why did you run the benchmarks without optimization? [...]

Optimization could remove or rearrange parts of code which would not happen in a real-life case. For example doing something like

```
for I in 1..1000 loop
   N := I;
end loop;
```

could be optimized to N := 1000.

IMO, not optimized code is a better measure for algorithmic complexity.

> [...]

> pragma journal_mode=WAL;

> pragma synchronous=OFF;

> [...]

Thanks for pointing this out.

Regarding Ada, it was strictly Ada.Direct_IO, nothing else. Ada.Direct_IO does not have Flush[*]. As far as I can tell GNAT's implementation of Ada.Direct_IO.Write is fwrite not followed by fsync. So forcing SQLite to sync might be unfair. However, the intended use surely must sync upon commit.

[*] Maybe it is worth an AI to add Flush to Direct_IO.

Data-structure Benchmark

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Wed, 17 Sep 2014 18:15:15 -0700
Subject: B-Tree v Skip-List Insertion Benchmark
Newsgroups: comp.lang.ada

I ran a quick comparison of the insertion times for Kazakov's Generic_B_Tree pkg against PragmARC.Skip_List_Unbounded from the PragmAda Reusable Components.

Both data structures are used for similar purposes, allowing O(log N) look-up times. Insertion and deletion are expensive operations on balanced trees due to re-balancing, and one reason skip lists were invented was to have fast insertion and deletion times compared to balanced trees. I had never actually compared times, and Kazakov's recent post comparing DB times got me thinking about it.

A typical run inserting the same 1,000 items into both structures gives times of

1.484 ms for the B tree

0.709 ms for the skip list

(Divide by 1,000 for average per-insertion times.)

The trade off is similar to using heap sort or quick sort. Both are O(N log N), with quick sort usually being faster. Heap sort is always O(N log N), but in rare cases, however, quick sort has worst-case performance of O(N**2).

A skip list is probabilistically balanced, and has a worst-case search time of O(N) in very rare circumstances, almost always for small N (< 256).

[See also “PragmAda Reusable Components”, AUJ 35-3, p. 154. —sparre]

Gnoga

From: David Botton <david@botton.com>
Date: Tue, 23 Sep 2014 11:20:39 -0700
Subject: Gnoga - The GNU Omnificent GUI for Ada
Newsgroups: comp.lang.ada

This is not an announcement of a 1.0 yet, but a progress report. It helps me stay focused and motivated along with a source for ideas and inspiration to post things, so here we go.

BTW, you can play the snake game running in Ada now over the internet, http://www.gnoga.com - NO JS and HTML that is done with Gnoga bindings to the Canvas and DOM.

I am busy working on the code at the moment so the documentation is mainly in the specs and samples for the moment and I have not had time to make a nice website for it yet, however http://www.gnoga.com is there and a link to the sourceforge site.

An introduction to what it is:

1. First and foremost the project goal is a cross platform GUI toolkit. but Instead of targeting Windows, X, Gtk or Qt, it targets the HTML5 browser. No not HTML or JS, the "browser". It acts like a "terminal" for Gnoga to render its magic. If you try and do a view source on the browser all you will get is the websocket code used to set up the communications. Long term you will be able to package a native app (.exe, .app, etc.)
2. Because the "browser" is the target it means that Gnoga applications can run local or remote. If you can get AWS running on your "board" you can use Gnoga for the front end. Yes you could write HTML and respond to HTTP requests etc using AWS, but with Gnoga your app is always connected and live in the browser back to the server. You can be showing real time stats, no Ajax, JSON, etc to worry about, oh yes and all of it is in Ada you don't have to touch those sick little braces { } pocked with ;;;;
3. It just happens to be that Gnoga can also create great websites with dynamic content using HTML, CSS and Ada (that's right not JS)... It's a nice bonus.
4. In fact Gnoga comes already with a number of Ada on Rails like features. Including Active Record support with bindings to MySQL and SQLite and can easily be expanded to other SQL engines.
5. There is a whole lot there already but.. there is still a lot more to go. Don't pass judgment till at least all the components are in, 6-8 weeks.
6. With this you will get things like OpenGL programming via WebGL (coming), it already has a full canvas 2d binding, and any other techi goodness thrown at the web.
7. Multimedia bindings are not far behind for video, audio, etc.
8. You will get access to client side HTML5 goodness like local storage on the client browser, etc. it's all coming.
9. Gnoga can easily use now or be extended later to bind anything that can run in a browser, JS GUI toolkits, XUL for direct native apps, etc. etc.
10. While most of the world is fighting to get JS and HTML to run and do anything, Ada get's to sit back and enjoy the ride to every new tech as it comes and still have a solid language and the ability to create secure Web apps and services dispatched from solid systems.

It's tough to get the full vision in words and there are not too many pictures to see, but if I get you excited about Ada for application development, well than I'm getting somewhere and we will both arrive soon enough at the goal :)

From: David Botton <david@botton.com>
Date: Sun, 28 Sep 2014 19:42:28 -0700
Subject: Re: Gnoga - The GNU Omnificent GUI for Ada
Newsgroups: comp.lang.ada

Today's update:

1) I have made numerous fixes to make sure the Ada code is compliant with my Ada coding standards.

2) I've made a number of bug fixes

3) I added a Console like View type with auto scrolling as elements added to bottom

4) I've added the first 2 tutorials on how to code in Gnoga: http://sourceforge.net/p/gnoga/code/ci/master/tree/tutorial/

5) I've made the audio and video types functional, although they need some more specific events and properties.

6) I added local client side storage support and session support based on sessionStorage.

From: David Botton <david@botton.com>
Date: Mon, 29 Sep 2014 20:31:35 -0700
Subject: Re: Gnoga - The GNU Omnificent GUI for Ada
Newsgroups: comp.lang.ada

Today's Updates :)

1) 2 more Tutorials

2) Views will now deallocate dynamically created child objects on finalization

3) Moved Gnoga.Application.Multiuser to Gnoga.Application.Multi_Connect

4) Modified how app data is set for connections to now use the Main_Window, it will also deallocate it if dynamically created on finalization.

5) It is no longer necessary to use Connection.Hold unless desired for clean up on connection events.

In general as I am writing the tutorials I am doing as much as possible to simplify the API and make coding easier in Gnoga.

Here are a list of planned tutorials so far (the first 4 are now done and in the repo)

Tutorial-01 - Introduction to Gnoga applications

Tutorial-02 - Introduction to Event Handlers

Tutorial-03 - Introduction to Multi-Connection Apps

Tutorial-04 - Tasking and Gnoga

Tutorial-05 - Using the Canvas Control

Tutorial-06 - Popups windows, iFrames, and custom boot files with Gnoga

Tutorial-07 - Forms and Gnoga

Tutorial-08 - Database bindings and Schema Migrations using Gnoga

Tutorial-09 - Active Record - Data modeling in Gnoga

Tutorial-10 - Creating MVC apps and Sessions management in Gnoga

In each tutorial directory there is a README that summarizes additional aspects of Gnoga learned in that tutorial. It is worth reading through the READMEs and sources in each tutorial in order as they build on each other. They also teach far more than just their subject line about things you can do with Gnoga.

From: David Botton <david@botton.com>
Date: Tue, 30 Sep 2014 17:22:27 -0700
Subject: Re: Gnoga - The GNU Omnificent GUI for Ada
Newsgroups: comp.lang.ada

So far for today added:

1) Ability to remove event handlers by setting to null

2) Corrected some bugs

3) Added Tutorial 05 - A quick little canvas drawing application to demonstrate the canvas and mouse events.

From: David Botton <david@botton.com>
Date: Sun, 12 Oct 2014 23:05:36 -0700
Subject: Re: Gnoga Latest Updates
Newsgroups: comp.lang.ada

Tutorial 09 Added

Learn about:

1) Interactive Forms

2) Tabs and the Card View

3) Using the Docker view for layout

From: David Botton <david@botton.com>
Date: Sat, 18 Oct 2014 21:40:02 -0700
Subject: Re: Gnoga Latest Updates
Newsgroups: comp.lang.ada

Tutorial 10 added

Illustrates:

1) Database bindings in Gnoga

2) Use of database migrations

From: David Botton <david@botton.com>
Date: Sun, 19 Oct 2014 16:02:39 -0700
Subject: Re: Gnoga Latest Updates
Newsgroups: comp.lang.ada

As of this last update:

1) I have added a simple all Ada template parser (so now possible to use PHP, Python or a simple token replace for text parsing)

2) It is no long required that you cd in to the bin directory to execute a gnoga application

3) The executable can be in a bin subdirectory or at the application root directory

4) Any missing sub directories (/js, /img, /css) are assumed to be in /html, if /html is also missing all files are assumed to be in the applications root directory. (e.g. you could place the snake executable and boot.html in the same directory and snake will run with no issue now)

From: David Botton <david@botton.com>
Date: Sun, 19 Oct 2014 21:36:02 -0700
Subject: New Gnoga Tool - gnoga_make
Newsgroups: comp.lang.ada

It's now even easier to write Gnoga apps with a new tool that is part of Gnoga - gnoga_make

Gnoga_Make works on Mac, Linux and Windows.

Gnoga_Make currently creates only one type of scaffolding for Gnoga apps a multi_connect app. There will be many more added before 1.0 in the next few weeks. (BTW, these scaffold apps also demonstrate good methods for developing Gnoga apps)

Example use:

Install Gnoga:

```
git clone
git://git.code.sf.net/p/gnoga/code
gnoga-code

cd gnoga-code

make install
```

(if on Mac / Unix and needed sudo make install)

This will build and install Gnoga as a standard gnat package and install gnoga_make in gnat/bin

With gnat/bin on your command line:

```
gnoga_make new My_New_App
        multi_connect
```

This will create a directory called my_new_app and create all the need files for a gnoga multi_connect application including makefiles, project files, etc.

```
cd my_new_app

make

bin/my_new_app
```

From: David Botton <david@botton.com>
Date: Thu, 23 Oct 2014 23:07:31 -0700
Subject: Cairo Bindings now added to Gnoga
Newsgroups: comp.lang.ada

I've adopted the Cairo bindings from GtkAda for Gnoga.

This adds in a quick instant tons of functionality for vector graphics. This of course is a great fit since Cairo will produce SVG in addition to PNG and PDFs so a really great fit.

I'll be adding a thicker layer to it for easier use in general and for Gnoga.

Cairo libs are usually installed already on Linux, for Mac I use home brew - brew install cairo and brew install libsvg-cairo for Windows install GtkAda even though not dependant on it, it installs all the needed libs for cairo.

Permutation Generators

From: jpwoodruff@gmail.com
Date: Fri, 3 Oct 2014 11:50:30 -0700
Subject: Re: Permutation generator in ada library
Newsgroups: comp.lang.ada

> Does Ada have a built-in function that given an integer N creates all possible permutations.

> I found this <http://rosettacode.org/wiki/Permutations#The_generic_package_Generic_Perm>, but was wondering if I can just call a built-in function?

I can address the original issue about permutation-generating Ada.

One is contained in the library Charles built by Matthew Heany: http://home.earthlink.net/~matthewjheaney/charles/index.html

His last update was in 2004. The materials are at: http://charles.tigris.org/source/browse/charles/src/

The second is by Mats Weber. My copy carries dates to 1990. Mats Weber's Ada Component Library, version 2.0: http://mats.weber.org/ada/mw_components.html

His document says:

Copyright (c) 1999 Mats Weber, Ch. du Grillon 10, 1007 Lausanne, Switzerland. These components were originally developed by Mats Weber at EPFL (Swiss Federal Institute of Technology, Computer Science Theory Laboratory and Software Engineering Laboratory) from 1985 to 1990

They carry the GNU General Public License.

My oldest holding is an archeological remnant from Simtel 20, built by Doug Bryan.

"This software is released to the Public Domain" but I don't know where there is a public copy. I'd be happy to share with anyone interested.

```
-- Unit name : Permutations_Class
-- Version : 1.0
-- Author : Doug Bryan
--           : Computer Systems Lab
--           : Stanford University
--           : Stanford CA, 94305
-- DDN Address : bryan@su-sierra
-- Copyright : (c) -none-
-- Date created :  15 April 1985
-- Release date :  15 April 1985
-- Last update :  15 April 1985
-- Machine/System Compiled/Run on :
-- DG MV/10000 ADE 2.2

generic
   type Item_Type  is private;
   type Index_Type is (<>);
   type List_Type  is array
        (Index_Type range <>) of Item_Type;

package Permutations_Class is

   generic
         with procedure Process
               (A_Permutation : List_Type);
   procedure Iterate_Through_Length_
         Factorial_Permutations
                     (Of_Items : List_Type);

   -- For an actual parameter for Of_Items
   --  of length n, n! (n factorial)
   -- permutations will be produced.
   -- The procedure permutes the elements
   -- in the array ITEMS.
   -- actually it permutes their indicies and
   -- re-arranges the items within the list.
   -- The procedure does not care of any or all
   -- of the items in the list are equal
   -- (the same).

end Permutations_Class;
```

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Sat, 4 Oct 2014 18:06:58 +0000
Subject: Re: Permutation generator in ada library
Newsgroups: comp.lang.ada

The code by Doug Bryan, that John mentions at the end of his posting, was included in the "Ada and Software Engineering Library Version 2 (ASE2)".

Numerous versions of the ASE library were put together by Richard Conn, the last one in October 2000. They were typically distributed on CDROM at the time, among others at various Ada events such as ACM SIGAda and Ada-Belgium conferences.

The last ASE2 version is still available on the Ada-Belgium site: ftp://ftp.cs.kuleuven.be/pub/Ada-Belgium/ase/index.htmThe s.c. "asset" that includes the Permutations_Class package is at: ftp://ftp.cs.kuleuven.be/pub/Ada-Belgium/ase/support/cardcatx/csparts.htm

The relevant source code is included in the files CSPARTS.SRC and CSPARTB2.SRC in the csparts.zip archive, retrievable via the above URL.

JSON Serialisation

From: Maxim Reznik <reznikmm@gmail.com>
Date: Fri, 24 Oct 2014 05:27:53 -0700
Subject: ANN: Serialization Ada objects into/from JSON
Newsgroups: comp.lang.ada

Now Matreshka provides support for serialization Ada objects into/from JSON format using 'Read/'Write attributes.

No magic involved. Conversion routines are provided by user with help of handy framework.

See an example http://forge.ada-ru.org/matreshka/wiki/League/JSON/Streams

[See also "Matreshka", AUJ 35-1, p. 8. —sparre]

SparForte

From: Ken Burtch <koburtch@gmail.com>
Date: Fri, 24 Oct 2014 04:18:42 -0700
Subject: ANN: Sparforte 1.5.1
Newsgroups: comp.lang.ada

This version fixes the fatal exception when loading include files ("with separate").

The source code is available on the website at

http://www.sparforte.com

[See also "SparForte", AUJ 35-3, p. 158. —sparre]

Request: GNAT for OpenVMS/Alpha

From: Eugen Wintersberger <eugen.wintersberger@gmail.com>
Date: Sat, 25 Oct 2014 09:35:32 -0700
Subject: Ada on openvms for Alpha
Newsgroups: comp.lang.ada

I have a rather unusual problem: I am looking for an Ada compiler for OpenVMS for Alpha. GNAT no longer supports OpenVMS for Alpha (a decision I Can entirely understand from an economical point of view). However, I Have a couple of Alpha boxes running OpenVMS and I would love to see them running Ada code.

Does anyone of you own a GNAT license for OpenVMS Alpha or knows someone who does and would be willing to give away this license or sell it to me?

Thanks in advance and best regards

AVR-Ada

From: Rolf Ebert <rolf.ebert.gcc@gmx.de>
Date: Sun, 26 Oct 2014 10:17:29 +0100
Subject: open issues for V1.3
Newsgroups: gmane.comp.hardware.avr.ada

[Preparations for AVR-Ada 1.3 release. —sparre]

I'd also like to include AvrX in a V1.3 and drop avr-threads, but I don't know when AvrX will be ready for AVR-Ada.

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Fri, 31 Oct 2014 22:11:15 +0200
Subject: Re: open issues for V1.3
Newsgroups: gmane.comp.hardware.avr.ada

[...]

My very unofficial build service seems to be able to build the repository now:

http://build.ada-language.com/job/AVR-Ada_Debian_7/

[See also "AVR-Ada", AUJ 34-2, p. 66. —sparre]

Ada-related Products

Status of Ada 2012 Implementations

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 14 May 2014 16:37:39 -0500
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

> Who else besides AdaCore is doing an Ada 2012 implementation?

Sadly, don't know of any. I've added a tiny amount of Ada 2012 stuff to Janus/Ada, but it will be a long time before much significant gets there.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Wed, 14 May 2014 17:56:33 -0400
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

> [...]

I'd bet Atego and ICSC are working on it.

CodePeer Earns Qualification for Software Verification in Avionics and Railway

From: AdaCore Press Center
Date: Thu Oct 23 2014
Subject: AdaCore's CodePeer Static Analysis Tool Earns Qualification for Software Verification in Avionics, Railway
URL: http://www.adacore.com/press/codepeer-earns-qualification/

Automatic code review and validation tool meets rigorous industry software verification standards; provides trusted reliability for Ada developers in safety-critical applications

NEW YORK, PARIS and BRISTOL, October 23, 2014, High Integrity Software Conference, Bristol, UK -- AdaCore today announced that its CodePeer advanced static analysis tool for the automated review and validation of Ada source code has been qualified as a software verification tool for developers in both avionics and railway industries.

CodePeer assesses the program before execution to find errors efficiently and early in the development life cycle. Using advanced mathematics, CodePeer analyzes every line of software, considering every possible input and every path through the program. It performs impact and vulnerability analysis when existing code is modified, and, using control-flow, data-flow and other advanced static analysis techniques, it detects problems that would otherwise require labor-intensive debugging.

"In safety-critical domains, developers need very strong assurances that the tool they're using to assess their code is reliable, can be trusted, and will substantially reduce the need for manual code review," says Arnaud Charlet, CodePeer Product Manager and Technical Director at AdaCore. "CodePeer has been through rigorous industry-specific tests for avionics and railway that fully affirm its value and reliability in these and other safety-critical development environments."

<u>Avionics Qualification</u>

CodePeer has been qualified as a verification tool for DO-178B, the software safety standard for commercial airborne systems. Certification authorities such as the FAA in the U.S. and EASA in Europe apply DO-178B to provide confidence that the software will meet its requirements.

Vulnerabilities detected by CodePeer analysis for avionics include following:

- Overflow on integer and floating point types
- Range violations on integer and floating point types
- Index violations on array operations
- Division by zero on integer and floating point types
- Uninitialized variables
- Underflow on floating point types

Where no potential error is reported, CodePeer guarantees that the code is exempt from these vulnerabilities

<u>Railway Qualification</u>

For railway applications, CodePeer has been used to verify code certified in accordance with CENELEC EN 50128:2011 SIL 4 --the highest safety integrity level.

In this context, CodePeer has been used for the following activities:

- Boundary value analysis: it detects attempts to dereference a pointer that could be null, to read values outside the bounds of an Ada type or subtype, and also detects buffer overflows, numeric overflow or wraparound, and division by zero.
- Control flow analysis: it detects suspicious and potentially incorrect control flows, such as unreachable code, redundant conditionals, loops that either run forever or fail to terminate normally, and subprograms that never return.
- Data flow analysis: it detects suspicious and potentially incorrect data flows, such as variables read before they are written (uninitialized variables), variables written more than once without being read (redundant assignments), variables that are written but never read, and parameters with an incorrect mode (unread parameter, unassigned parameter).

CodePeer can be used in conjunction with AdaCore's GNAT Pro development environment where it is tightly integrated into AdaCore's GPS (GNAT Programming Studio) and GNATbench IDEs, or as a standalone product. It comes with a number of complementary static analysis tools common to the technology: a coding standard verification tool (GNATcheck), a source code metric generator (GNATmetric), a semantic analyzer and a document generator.

A demo highlighting the new features introduced in the latest version of CodePeer can be viewed at the following url: http://www.adacore.com/codepeer-2-3-demo/

[See also "CodePeer", AUJ 35-1, p. 10. —sparre]

Ada and Operating Systems

Fedora: GtkAda

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>
Date: Thu, 24 Jul 2014 10:38:44 +0200
Subject: GTKada 3 is in Fedora
Newsgroups: gmane.comp.gnome.gtk+.ada
To: gtkada@lists.adacore.com

For anyone who is interested: GTKada 3.8.2 is now packaged in Fedora. The package is named "GtkAda3".

Version 2.24.2 is still available as "GtkAda". The binary libraries are parallel-installable, so programs using GTKada 3 can coexist with programs using GTKada 2. The -devel packages conflict though, because they use the same filename in several cases, so you can develop for GTKada 2 or for GTKada 3, but not both simultaneously.

MacOS X: XNAdaLib

From: Pascal Pignard <p.p11@orange.fr>
Date: Mon, 08 Sep 2014 18:17:12 +0200
Subject: [ANN] XNAdaLib 2014 binaries for MacOS 10.9 including GTKAda 3.8 and more.
Newsgroups: comp.lang.ada

This is XNAdaLib 2014 built on MacOS X 10.9 Mavericks for Native Quartz including:

- GTK Ada 3.8.2 with GTK+ 3.10.7 complete for Quartz backend,
- Glade 3.16.1,
- GnatColl GPL 2014,
- Florist GPL 2014,

- AdaCurses 20110404 (http://invisible-island.net/ncurses/ncurses-Ada95.html),
- Gate 3-04-b (http://sourceforge.net/projects/lorenz),
- AICWL 3.9 (http://www.dmitry-kazakov.de/ada/aicwl.htm with Components 4.1 and gtksourceview 3.10.1),

to be installed (mandatory) at /usr/local:

```
$ cd /usr/local
$ sudo tar xzf xnadalib-gpl-2014-quartz-x86_64-apple-darwin13.3.0-bin.tgz
```

Update your PATH to include gtkada-config, glade, gate3.sh and other executables in it:

```
$ PATH=/usr/local/xnadalib-2014/bin:$PATH
```

Update your GPR_PROJECT_PATH to include gtkada.gpr, adacurses.gpr, florist.gpr, gnatcoll.gpr, gtkada_aicwl.gpr and other projects in it:

```
$ export GPR_PROJECT_PATH=/usr/local/xnadalib-2014/lib/gnat:$GPR_PROJECT_PATH
```

Set XDG_DATA_DIRS for GNOME apps:

```
$ export XDG_DATA_DIRS=/usr/local/xnadalib-2014/share
```

Glade and GPS applications in apps directory must stay in this directory unless you modify the script inside apps.

Then see READMEs, documentation and examples in share directory and enjoy.

XNAdaLib binaries have been post on Source Forge:

http://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2014-mavericks/

The instructions for building XNAdaLib are here:

(French language)

http://blady.pagesperso-orange.fr/telechargements/gtkada/Install-GTKAda-Quartz.pdf

Feel free to send comments.

Debian: SQLite Interface

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 07 Oct 2014 01:35:07 +0200
Subject: Re: Which database document for wheezy?
Newsgroups: gmane.linux.debian.packages.ada

> [...]

Yes, there is GNADE, the ancestor of gnatcoll for SQLite connectivity.

aptitude install libgnadesqlite3-2-dev

I'm afraid there is no textbook on how to use GNADE, you'll have to read the Ada specs.

Debian/Windows: GNAT

From: Brian Drummond <brian@shapes.demon.co.uk>
Date: Thu, 09 Oct 2014 18:04:30 GMT
Subject: Re: Newcomers to comp.lang.ada: welcome and how did you end up here ?
Newsgroups: comp.lang.ada

> [...]

On the subject of mingw, I don't know how many people know of this option, but I was a little surprised to see mingw packages, including FSF GNAT, available on Debian.

Turns out it's a cross-compiler. So having developed an Ada app on Debian, I can invoke the mingw crosscompiler and build a Windows executable. So far these have worked flawlessly, including interfacing between Ada and a C library talking to a USB device.

The executable is larger - typically 800k instead of 150k for native Linux executables. Haven't investigated why but I assume it's statically linked to eliminate dependencies, and I haven't had to install anything other than the exe on Windows machines so far.

One more option and probably the simplest way to use FSF GCC targetting Windows machines...

Raspbian: Gnoga

From: Tony G. <tonythegair@gmail.com>
Date: Wed, 22 Oct 2014 09:24:07 -0700
Subject: Gnoga, raspbian jessie and the PI
Newsgroups: comp.lang.ada

I don't know if anyone else has tried, but I have just successfully built GNOGA and the tutorials successfully on a Raspberry Pi with the standard issued Debian packages AWS 3.2 and gnat (don't know the version).

Raspbian version is Jessie.

Mac OS X: GCC

From: Simon Wright <simon@pushface.org>
Date: Sat, 25 Oct 2014 20:30:07 +0100
Subject: ANN: GCC 4.9.1 for Mac OS X Mavericks and Yosemite
Newsgroups: gmane.comp.lang.ada.macosx

It occurs to me that I should probably have been making these announcements here as well as in c.l.a. Apologies to those of you who're already aware.

GCC 4.9.1 is available at https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/4.9.1

It was built on Mavericks and is compatible with Yosemite.

The README:

This is GCC 4.9.1 built for Mac OS X Mavericks (10.9.5, Darwin 13.5.0), with Xcode 6.0.1.

gcc-4.9.1-x86_64-apple-darwin13.tar.bz2

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 233185 (XMLAda-SVN for short).
- AWS from the public git repository[3] at commit e0d260e2d5dbbd935779493079 35848de2390818 (AWS-git for short).

Target: x86_64-apple-darwin13

```
Configured with: ../gcc-4.9.1/configure \
 --prefix=/opt/gcc-4.9.1 \
 --disable-multilib \  --disable-nls \
 --enable-languages=
 c,c++,ada,fortran,objc,obj-c++ \
 --host=x86_64-apple-darwin13 \
 --target=x86_64-apple-darwin13 \
 --build=x86_64-apple-darwin13 \
 --with-host-libstdcxx=-lstdc++
Thread model: posix
gcc version 4.9.1 (GCC)
```

MD5 (gcc-4.9.1-x86_64-apple-darwin13.tar.bz2) = f04d5d773174a4a58cdd2dd4871785a4

[1] http://www.gnu.org/licenses/gcc-exception-faq.html

[2] http://svn.eu.adacore.com/anonsvn/Dev/trunk/xmlada

[3] http://forge.open-do.org/anonscm/git/aws/aws.git

Debian: Adabrowse

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 27 Oct 2014 00:32:30 +0100
Subject: Re: Upload of adabrowse
Newsgroups: gmane.linux.debian.packages.ada

> [...]

OK, uploaded to unstable.

References to Publications

20 Years of Industrial Theorem Proving with SPARK

From: Roderick Chapman and Florian Schanda
Date: Tue Jul 15 2014
Subject: Are We There Yet? 20 Years of Industrial Theorem Proving with SPARK
URL: http://proteancode.com/keynote.pdf

This paper presents a retrospective of our experiences with applying theorem proving to the verification of SPARK programs, both in terms of projects and the technical evolution of the language and tools over the years.

[...]

Ichbiah's Resignation Letter

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Fri, 24 Oct 2014 11:47:08 -0700
Subject: Re: Ichbiah's Letter
Newsgroups: comp.lang.ada

> [...]

It is at

https://duckduckgo.com/l/?kh=-1&uddg=http%3A%2F%2Fweb.elastic.org%2F~fche%2Fmirrors%2Fold-usenet%2Fada-with-null

Ada Inside

Drop-in Ada (SPARK) Components

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Tue, 22 Apr 2014 14:18:44 +0200
Subject: Re: OpenSSL development (Heartbleed)
Newsgroups: comp.lang.ada

Alan Browne wrote:

> Is it possible to identify a particular client side layer item (app, transport, internet or link) that is relatively small that could be designed and written in Ada and that could "drop in" as a replacement?

> Obviously it would have to hook up and down in the system and 'look' for all intents and purposes like its C predecessor?

One such example is the Ironsides DNS server, I think,

http://ironsides.martincarlisle.com/

I guess the program may well be a target for appraisal. In any case, since this can replace one layer item, it is proof of concept.

Would people at Cisco take note of the possibilities of "language advantages", and S/E? (If they are "allowed" to make their devices more secure, which I do not know.)

http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20140409-heartbleed

Another hint is found in the use of Ada when cracking the Lorenz code. According to the winner, the cryptographic algorithms were expressed more clearly, and, quoting, *concisely*!

http://www.drdobbs.com/parallel/tunny-colossus-and-ada-keeping-an-open/207800151

[See also "Authoritative DNS Server", AUJ 34-3, p. 146. —sparre]

AVR-Ada in Hobby Projects

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Fri, 27 Jun 2014 09:14:53 +0300
Subject: Usage of AVR-Ada (Was: Lcd and arduino nano)
Newsgroups: gmane.comp.hardware.avr.ada

Rolf Ebert wrote:

> Nice to see that someone still uses AVR-Ada.

Almost every month someone emails me (or communicates via some other channel) and tells that they are using AVR-Ada and happy to read my Arduino blog (arduino.ada-language.com). So there are users, they are just little bit shy and don't discuss in public.

Personally, I have been busy with other projects, so I haven't had time to commit anything to AVR-Ada repo lately, but I am also using it.

For example, I have had AVR-Ada based wireless temperature sensor running on my balcony almost one month (the device is Olimexino-328 with custom XBee shield, powered by single 1000mAh lipo).

I plan to write about it, but I am still waiting for the battery to run out - not sure how many weeks I need to wait. :)

From: Jerry Petrey <gpetrey@earthlink.net>
Date: Fri, 27 Jun 2014 09:09:50 -0700
Subject: Usage of AVR-Ada (Was: Lcd and arduino nano)
Newsgroups: gmane.comp.hardware.avr.ada

Rolf and Tero,

I too am a happy user of AVR-Ada. As a long time Ada programmer in my professional career, I am very pleased to have Ada available on micros like the AVR for my hobby projects. You guys have done a great job and I hope you can continue to support it. More people need to discover the beauty of Ada for environments like these.

Thanks again for your efforts. I look forward to more great things from you.

New Spanish Satellite Project

From: AdaCore Press Center
Date: Thu Oct 23 2014
Subject: AdaCore Development Environment Selected for New Spanish Satellite Project
URL: http://www.adacore.com/press/spanish-satellite-project/

NEW YORK, PARIS and BRISTOL, October 23, 2014, High Integrity Software Conference, Bristol, UK – AdaCore today announced that its GNAT Pro cross-development environment has been selected by the Polytechnic University of Madrid (Universidad Politécnica de Madrid / UPM), for the UPMSat-2 UNION satellite project's real-time on-board and ground control software. The 50kg micro-satellite, scheduled to be launched in Q4 2015, will provide a technology demonstration platform for the university from a sun-synchronous orbit nearly 600 km above Earth.

The software component of the project is being led by UPM's Real-Time Systems and Telematic Services Engineering Research Group (Grupo de Sistemas de Tiempo Real e Ingeniería de Servicios Telemáticos / STRAST), with coding and testing scheduled to be completed by the end of 2014. The development environment is GNAT Pro for 32-bit Linux, targeted to the LEON3 processor.

UPM STRAST selected Ada for its combination of high reliability, speed of development, and ease of verification and validation. The team has extensive experience using Ada on previous high-integrity embedded system projects, and has collaborated with AdaCore on a number of these.

The STRAST team is using the GNAT technology to program the control software of the satellite's on-board LEON3 processor, which is expected to reach 20,000 lines of code. UPM STRAST is using its own Open Ravenscar Real-Time Kernel (ORK), along with the Ada code generator from the TASTE toolset (The ASSERT Set of Tools for Engineering). AdaCore verification and validation tools will be used to ensure code integrity, using an approach based on the ECCS-ST-E40 standard.

"Controlling a satellite's operation requires software that meets the highest levels of reliability and integrity," said Professor Juan Antonio de la Puente, Universidad Politécnica de Madrid. "Ada was the obvious choice, and the combination of GNAT Pro and model-based code generation has proved to be a very fast way of developing reliable software for this project. UPMSat-2 provides the perfect platform for our students to develop their skills, and for STRAST to demonstrate its capabilities in real-time embedded systems to potential commercial partners."

The UPMSat-2 hardware platform's on-board computer is an ACTEL FPGA board developed by TECNOBIT, with UPM responsible for synthesizing the System On Chip (SOC) from the Gaisler GRLIB IP Library for LEON3 processors.

"We have collaborated with the team at UPM STRAST for 20 years across a number of projects," said Cyrille Comar, AdaCore Managing Director. "This ambitious satellite project demonstrates all the advantages of Ada as a language, requiring reliable, real-time software that will need to operate in the toughest conditions. We look forward to the successful completion of the satellite and its launch in 2015."

Ada in Context

Wish-list: Huge Integer Literals

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 4 Apr 2014 15:53:44 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

>> I : Unbounded_Integer := +1E1000;

> I would like to see the definition of "+".

I missed that someone wrote a literal that's insanely large. They should simply have written:

```
I : Unbounded_Integer :=
        (raise Storage_Error);
```

because that's what will happen. I was thinking about more realistic cases:

```
Thousand : Unbounded_Integer := +1000;
```

But anyway, what works today for this *exact* literal (and not the more realistic cases I was thinking about):

```
Really_Large : Unbounded_Integer
        := +10**1000;
```

alternatively:

```
Really_Large : Unbounded_Integer
        := Value ("1E1000");
```

[since you're going to have Image and Value routines anyway].

"+" looks like:

```
type Largest_Int is range
        System.Min_Int .. System.Max_Int;
function "+" (Right : Largest_Int)
        return Unbounded_String;
```

I have an 64-bit math package for Janus/Ada that works exactly this way (need to it deal with some returns from OS operations), and it works well. Most of the literals that are needed are small (0, 1, 2, 10) and it's much preferable to write them using "+" rather than some unwieldy function name (To_Huge_Integer?).

One could do something similar with Value if large literals were really common, but I doubt that they'll appear in expressions very often.

Wish-list: Unary Type Conversion Operator

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 4 Apr 2014 15:43:17 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

J. Kimball wrote:

> On the other hand, using "+" operators in these ways obscures it's real meaning to people and often the compiler. In using a system that combines renames of language-defined and Templates Parser-defined conversion functions things can get rather hairy. Object names usually indicate what's being described, but rarely what type it is. A rich use of the type system makes using "+" boorish.

I find this attitude infurating. (And it's wrong, too; literals provide no useful type information and adding "+" to the front does not change that situation. Taken literally and to the extreme, your thinking implies that all overloading of operators is a bad thing because it obscures the types involved. But let's stick with the infurating part). Let me give you a bit of history:

Very early in the design of Ada, there was a proposal to add a unary operator symbol specifically for the purpose converting between types. Ichbiah and his team rejected the proposal as "+" already exists and has no other useful purpose. They said that "+" should be used for this purpose.

The idea to add a unary operator symbol resurfaces periodically, but it always gets shot down because "+" works for that purpose.

OTOH, attempts to actually *use* "+" in that way in the language-defined libraries also have always gotten shot down because there is a group which cannot stomach using it for non-numeric purposes. For instance, we had proposed to add:

```
function "+" (A : String)
        return Unbounded_String
        renames To_Unbounded_String;
```

to the Unbounded_String package because the conversion here is way too wordy. (Most of my packages that use unbounded string start with this declaration. The real problem is getting too many such declarations colliding.)

The net effect is that Ada has neither an explicit conversion operator nor the balls to use "+" as intended. Which makes using language-defined packages a wordy mess to the point that I try pretty hard to avoid them. That's not how that's supposed to work!

Attitudes such as yours prevent using the language as it was (and is) intended. And similar attitudes (on the other side of the debate) prevent changing it to make that less controversial. It leaves most people thinking the language has no way to do things when in fact the solutions have been there ever since the beginning of Ada.

As for the difficulty of figuring out errors in complex expressions -- remember two things: (1) quality of error handling is not something that the standard can changes; and (2) qualified expressions and prefix notation are your friend. Compilation is quick enough these days that there is no real problem sticking in some qualifications and/or prefix calls to narrow down problems in complicated expressions. (And why are you writing complicated expressions in the first place? Use some expression functions to break those up.)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 8 Apr 2014 18:44:12 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

J. Kimball wrote:

> Maybe they could agree on a new unused operator being added.

Did you read my original message? That's been suggested for many years. There is a camp that thinks "+" is good enough for that, and thus blocks any attempts to add another such operator. (There's a lot of people in that camp.)

The other group hates the idea of using "+" for that purpose, and blocks any use of that as a conversion in the language. (There's a lot of people in this group, too -- some are in both groups.)

The ARG operates by consensus. We have no consensus on either point, thus nothing gets done at all. (Luckily, this dynamic doesn't happen very often.)

> [...]

If it was up to me, '@' or '#' or '$' or '~' would have been used for this long ago.

> [...]

Probably any choice will annoy someone. I think that's the primary argument of the "+" backers -- no other solution is really obviously better, so let's not clutter the language further. It's hard to argue with that.

Fun With Specifications for Main Procedures

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Fri, 04 Apr 2014 15:58:05 -0700
Subject: Re: gnatmake error I don't understand
Newsgroups: comp.lang.ada

> Sure the ".ads" file may be extraneous, but won't hurt anything to my knowledge.

It might help. There's a little trick some colleagues played on people who left their workstations unlocked. Let us say Bob is working on a program called Alice and has the main subprogram in the file Alice.adb:

```
procedure Alice is
  ...
end Alice;
```

He has it to the point that he can run it, though some essential functionality is missing. We come along and create Foo.ads:

```
package Foo is
  pragma Elaborate_Body;

  exception Bar;
end Foo;
```

Foo.adb:

```
package body Foo is
  -- Nothum, eh?
begin -- Foo
  raise Bar;
end Foo;
```

and Alice.ads:

```
with Foo;
procedure Alice;
```

Now when he tests Alice, she raises Foo.Bar! People can spend a lot of time trying to figure that out.

Finding Unneeded “with” and “use” Clauses

From: Frank <dontspam365@gmail.com>
Date: Sun, 6 Apr 2014 13:56:31 -0700
Subject: Remove un-necessary “with” and “use”
Newsgroups: comp.lang.ada

Is there a way to remove/get warning about “with” and possibly “use” that are not “contributing to the executable”, via GPS 5.2.1 or possibly some of the GNAT tools?

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Mon, 07 Apr 2014 06:51:52 +0200
Subject: Re: Remove un-necessary “with” and “use”
Newsgroups: comp.lang.ada

> [...]

With AdaControl:

```
check unnecessary_use_clause;
check with_clauses (reduceable);
```

From: Simon Wright <simon@pushface.org>
Date: Mon, 07 Apr 2014 07:21:39 +0100
Subject: Re: Remove un-necessary “with” and “use”
Newsgroups: comp.lang.ada

> [...]

If you want to get the warning so that you can remove (or move) the “with”s, then -gnatwu will do the trick.

If you want most standard warnings, -gnatwa does it. If you want most standard warnings but not unused “withs”, use -gnatwaU.

Or you could say

```
pragma Warnings (Off);
with Unused_Package;
pragma Warnings (On);
```

in the source text.

A Language for Engineers

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 18 Apr 2014 19:30:00 +0200
Subject: Re: Heartbleed
Newsgroups: comp.lang.ada

> [...]

<rant>

The cause of Ada not being popular is that it has been designed to force people to THINK and do things cleanly. People prefer wild hacking and long debugging sessions to sitting back in one's chair and analyzing the problem.

</rant>

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Fri, 18 Apr 2014 11:04:26 -0700
Subject: Re: Heartbleed
Newsgroups: comp.lang.ada

> [...]

Yes. Or as I like to put it, Ada is a software-engineering language, and only 2% (in my experience) of developers are software engineers. The remaining 98% are not going to like Ada; they like hack-away languages like C.

Implementation Languages for Compilers

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 18 Apr 2014 12:42:34 +0200
Subject: Re: Heartbleed
Newsgroups: comp.lang.ada

Yannick Duchêne wrote:

> I personally see no requirement for an Ada compiler to be written in Ada. A statically typed and modular sufficiently high level language may be as much fine as Ada. There is no requirement of course, but some good reasons to write a compiler in its own language:

1) There is in general a commonality between a language, its representation, and the structures it handles best. Representing the language with its own structures is generally appropriate.

2) It makes porting the compiler to other machines easier (description of why is to be found in any good book about compilation)

3) Compiling the compiler with itself is an excellent test: the 2nd compilation should be identical to the third compilation, or there is something wrong...

The Strenghts of Ada

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sat, 19 Apr 2014 13:50:11 +0200
Subject: Re: Oberon and Wirthian languages (was: Heartbleed)
Newsgroups: comp.lang.ada

> [...]

The problem I have with Oberon and its descendants is that they removed the subrange types from Modula-2 (they are similar to Ada's subtypes of numeric types). Also, TTBOMK, no Wirthian language allows the programmer to define new numeric types from scratch and make them incompatible at compile-time (i.e. requiring explicit type conversion).

According to John McCormick's famous research paper[1], the most desirable features of a programming language are, in order of importance:

- Modeling of scalar objects.
  + Strong typing.
  + Range constraints.
  + Enumeration types.
- Parameter modes that reflect the problem rather than the mechanism.
- Named parameter association.
- Arrays whose indices do not have to begin at zero.
- Representation clauses for device registers (record field selection rather than bit masks).
- Higher level of abstraction for tasking (rendezvous rather than semaphores).
- Exception handling.

And personally, I share his opinion :)

So, Oberon-14 or whatever its name is should not only reinstate subranges but also allow the definition of incompatible scalar types. If it did support all of the desirable features above then it would effectively almost become Ada :)

Notable features absent from that list include generics, type extension, dynamic dispatching, subtypes of non-scalar types, nested subprograms and overloading. A subset of Ada omitting these features would require a compiler and run-time system much simpler than full Ada and still bring huge benefits to the safety of programming. Access types are required no matter what :/

[1] http://archive.adaic.com/projects/atwork/trains.html

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Sat, 19 Apr 2014 14:46:50 +0200
Subject: Re: Oberon and Wirthian languages
Newsgroups: comp.lang.ada

Ludovic Brenta wrote:

> Access types are required no matter what :/

Parasail[1] (and some other experimental languages, I think) seem to tackle pointing with the help of components marked "optional", accompanied by specially designed definitions for copying, moving, and swapping. This combination is said to prevent the dangers of pointers.

[1] http://parasail-programming-language.blogspot.de/2012/08/a-pointer-free-path-to-object-oriented.html

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Sat, 19 Apr 2014 18:53:48 +0200
Subject: Re: Oberon and Wirthian languages
Newsgroups: comp.lang.ada

Ludovic Brenta wrote:

> [...] And personally, I share his opinion:)

The most important finding that McCormick's list represents is that they are *not* an opinion! The evidence is one rare exception in that its production exhibits many traits of valid data.

The type system *is* actually better that that against which it has been compared. It is not just opined to be better.

Object'Image

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Wed, 23 Apr 2014 05:55:00 -0700
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> I wonder what is high on your list of wishes for Ada 202X?

I'd like V'Img to be standard ada like T'Image(V);

As in the gnat implementation

```
type T is some_discrete_type
V : T;
Put_Line(V'Img);
```

Also, I'd like to be able to define the string function for a record type to be used for the 'image attribute.

```
type T2 is record
  A : T;
  B : T;
end record;

function F_T2_Image(O : T2)
        return String is
begin
  return O.A'Img & " " O.B'Img;
end F_T2_Image;

for T2'Image use F_T2_Image

V2 : T2;
...
Put_Line(V2'Img);
```

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Wed, 23 Apr 2014 09:31:29 -0700
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> Ah! Another case of "I don't want to use the use clause, give me something else that avoids writing these damn long names".

No. using the 'use' is certainly something one can have different opinions on. But I like to avoid these kind of errors

SET_ERROR_MODE(NO_ERROR, CRANE_TYPES.ASSIGNMENT_TIMEOUT, FALSE); |

>>> error: "NO_ERROR" is not visible

>>> error: multiple use clauses cause hiding

>>> error: hidden declaration at crane_types.ads:113

>>> error: hidden declaration at siemens_interface.ads:374

>>> error: hidden declaration at core_types.ads:71

One of the best thing with Ada05 was the approval of object.verb notation. Even if i get to use it seldom at work, I do in hobby projects, just for this reason. The variable/object knows where it belongs, no need to use 'use' everywhere.

This is the basic idea for my 'img proposal. The variable knows its type. No need to have long package names or risk hidden declarations.

By the way, the above hidden situation was because of adding a constant 'NO_ERROR' in siemens_interface.ads

The other two was a constant and a coded value. No hiding before that.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 23 Apr 2014 18:42:55 +0200
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...]

I would tend to say that the "use" has the benefit to show you that you used the same name in various contexts with various meanings, and that a bit of reengineering might be in order...

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Wed, 23 Apr 2014 10:51:03 -0700
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...]

Yes. It might. Or not.

The code is written by several programmers during different times. I think each of them had good reasons to declare a constant NO_ERROR, when communicating with different devices, like PLCs. Either in a conveyor sub system or in a crane sub system. Calling them SIEMENS_S5_NO_ERROR or CRANE_NO_ERROR would put the ambiguity away, but the code would look awful.

The clash was recent, due to ever evolving changes - new demands from customers. New demands lead me to define yet another NO_ERROR constant, for a new subsystem, Siemens s7.

But reengineer a running project due to I cannot use 'use' in this context. Well, yes, if the customer pays for that. Otherwise, they are happy with me qualifying NO_ERROR with correct package name.

So, In a technical sense I agree. Re-engineer. But in a practical sense I do not. Cost way too much, and gains too little.

While my suggestions about 'Img may not gain very much, it would help at least me. And it would probably not cost very much.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 23 Apr 2014 15:14:11 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

Jeffrey Carter wrote:

> [...] desiring V'Image rather than T'Image (V) may not be entirely about saving keystrokes.

It's not, it's mainly about wanting to avoid the effort to look up the exact subtype of an object before writing 'Image. That wastes far more time than the few keystrokes ever would take. (Which is redoubled if one has to find out whether Image exists or some function has to be used instead.)

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Thu, 24 Apr 2014 02:16:32 -0700
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...]

I agree. It is not just to save keystrokes, it is to simplify work. And no-one here uses a debugger, so examining log-files is the way to find errors (around here anyway). Debugging does not help when you want to examine why something happened or did not happen at a running site.

Logfiles do. If they contain the correct amount of logging.

'Img or 'Image would encourage more people to log stuff they perhaps do not know that they need to log. And it is a pain if you want to log a record, with many separate types, where the types are defined in several separate files.

Variable'Img or Variable'Image would make life easier for me.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 23 Oct 2014 21:45:23 -0500
Subject: Re: Assembling Complex Strings Containing Carriage Returns Prior to Using Ada.Text_IO.Put?
Newsgroups: comp.lang.ada

> [...]

Ah, but Ada 2012.5 :-) will have Object'Image as a language-defined attribute. AI12-0124-1 was approved for inclusion in the upcoming Corrigendum at the recent Portland ARG meeting.

This is a case where we (the ARG) decided that including existing practice in the Standard made sense. (Note that we used 'Image for this purpose; AdaCore couldn't do that because extending language-defined attributes is prohibited, but the ARG has no such problems.)

When to Use "use" Clauses

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 23 Apr 2014 15:11:40 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

Jean-Pierre Rosen wrote:

> I would tend to say that the "use" has the benefit to show you that you used the same name in various contexts with various meanings, and that a bit of reengineering might be in order...

That's baloney. For instance, if we were to add an exception to Claw, it might very well have the same name as some exception the client (or even another third party) declared somewhere. Why do you think this is a problem? The teams maintaining the subsystems no contact and only happen to be both included in some client program. Our maintenance, however, could break the client program even though there is no intended use of the new entity. That's just wrong.

I've come to realize that the problem isn't use-clauses per-se, it's use clauses of things that maintenance can change (specifically when changes to non-overloadable entities can happen). As such, package use clause is acceptable on language-defined packages that don't allow implementation-defined identifiers, but that's it. In contrast, "use all type" (and the more limited "use type") are acceptable anywhere, as their maintenance hazard is much more limited, mostly to things for which having the same name and type profile is dubious anyway.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 23 Apr 2014 15:03:26 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

Björn Lundin wrote:

> One of the best thing with Ada05 was the approval of object.verb notation.

Ada 2012 adds "use all type" with essentially the same semantics as object.verb notation. If you have untagged types, especially enumerations, I strongly suggest using that. (Since it only makes overloadable entities visible, it doesn't have the maintenance hazard unless the profiles match -- in which case you have a design problem.)

Late Declaration of Names Used in Aspects?

From: Simon Wright <simon@pushface.org>
Date: Thu, 24 Apr 2014 11:12:55 +0100
Subject: Declaration of function in precondition
Newsgroups: comp.lang.ada

I proposed this in answer to a question[1] on StackOverflow:

```
package Ring_Buffer is

   function Is_Full return Boolean;
   procedure Push (Value : T)
         with Pre => not Is_Full;
   function Pop return T;

private
   Buffer   : array (0 .. Size) of T;
   Read_At  : Integer := 0;
   Write_At : Integer := 1;

   function Is_Full return Boolean
         is (Read_At = Write_At);
end Ring_Buffer;
```

and it turns out that GNAT (GPL 2013, 4.9-20140119) is happy if I put the spec of Is_Full after its use (but still in the visible part):

```
   procedure Push(value: T)
         with Pre => not Is_Full;
   function Is_Full return Boolean;
```

I can't see where in the ARM this is legalised?

[1] http://stackoverflow.com/questions/23203022/ada-aspects-which-are-private-to-a-package

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 24 Apr 2014 08:22:55 -0700
Subject: Re: Declaration of function in precondition
Newsgroups: comp.lang.ada

> [...]

13.1.1(11), I think. But I'm not sure.

"The usage names in an aspect_definition are not resolved at the point of the associated declaration, but rather are resolved at the end of the immediately enclosing declaration list."

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 24 Apr 2014 20:45:48 -0500
Subject: Re: Declaration of function in precondition
Newsgroups: comp.lang.ada

> [...]

Correct. In this case, aspect specifications are resolved at "private", so you can use anything in the visible part in them.

This property is necessary for some type-related aspects, else they would be useless:

```
type Priv is private
 with Read => Read,
       Write => Write,
       Type_Invariant => Is_Valid (Priv);
```

Since all of these need access to subprograms that have parameters of type Priv, and those *have* to follow the type declaration of Priv, none of these things could have been specified as aspects without this (admittedly strange) rule.

Most aspects are evaluated at the first freezing point of the associated entity (type in this case), so oddities are possible. There are some rules to prevent the worst ones -- but it's very unlikely that you'll ever run into them. After all, "the first freezing point" is something you worry about only if the compiler complains, and I recommend the same approach here. (Coincidentally, I was working on objectives and ACATS tests for those rules yesterday, so I'm more aware than usual about them.)

Safe Use of Mutexes

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 4 May 2014 21:34:22 +0200
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

[...] Mutex should always be handled by a controlled "holder" object:

```
declare
   Lock : Holder (Resource'Access);
   -- Seize the resource
begin
   Map.Find("Something");
end; --  Release the resource
```

This guaranties that the resource will be released even upon an exception propagation.

Regarding containers it is recommended to use reentrant mutexes if operations will be extended or if you fancy re-dispatching. An implementation of reentrant mutex can be found here:

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

Task Safety of “constant” Objects?

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sun, 04 May 2014 11:55:49 -0700
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

Natacha Porté wrote:

> I have some shared resources indexed by a string, described in a file, and considered as constant throughout the lifetime of the program (with the standard scheme "restart for changes to take effect").
>
> When I have whatever indexed by a string, I immediately think Maps, and usually go for Ordered_Maps because they are easier to understand.
>
> Since the map is semantically constant, I use the magic word constant, with a function to load from file at elaboration, and everything seems to work fine.

So, you have

```
M : constant Map := F;
```

and you're concerned about concurrent calls to

```
M.Element (Key)
```

The reserved word constant means that you can't assign to the object or pass it as an [in] out parameter. It certainly says nothing about what can happen to things designated by an access component of the object, and unbounded containers should be expected to have access components.

```
type AI is access Integer;
C : constant AI := new Integer'(1);
V : AI := new Integer'(2);
C := V; -- illegal
C.all := 42; -- No problem
```

As you've noted, the ARM says nothing about task safety for containers in general, maps in general, or ordered maps in specific, so you can't rely on this being task safe. And since you have the source to GNAT's ordered map package, you can look at it and see that this specific implementation is not task safe.

As you note, the standard allows for simultaneous calls to protected functions, so in general putting the map in a protected object and allowing access through a protected function doesn't gain you anything. Accessing it through a protected procedure, however, does guarantee non-concurrent access.

Since you have access to the source of GNAT, you can see that it locks a PO even for function calls, so with GNAT a protected function shouldn't be a problem. I've worked on a project that used GNAT and AWS and had many tasks accessing hashed maps in protected objects without problem.

If you're interested in a solution that is not GNAT-specific, the skip-list implementation in the PragmAda Reusable Components has a Search operation that is task safe. It's easy enough to use a skip list as an ordered map. You'd have to use a[n] [Un]Bounded_String for the key, but that shouldn't be too much of a problem.

The PragmARCs are available from

http://pragmada.x10hosting.com/pragmarc.htm

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 7 May 2014 22:19:12 -0500
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

> [...]

Ada really doesn't have any such thing as constant objects for many types. The majority of "constants" of composite types are actually variables during some part of their lifetime (and because that variable view can be saved and used later, they can never be assumed to be constant). That specifically applies to anything with a controlled part and anything with an immutably limited part.

As a client, since you shouldn't be looking through private types, you have to assume that there is a controlled component somewhere and thus you should never assume *anything* is constant of a private type.

Ergo the question is meaningless for the vast majority of constant composite objects; they exist in name only.

“Task_Safe” and “Potentially_Blocking”

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 6 May 2014 21:07:34 +0200
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

> What exactly do you mean by "task safe", [...]

It means, in my interpretation, that the post-condition of the operation [and the object's invariant] is true for any number of tasks Wi calling the operation independently at any point Ti of real-time.

> If both Element and Replace_Element are task safe, does that mean calls to Element and Replace_Element are atomic; i.e. if one task calls Element, and another calls Replace_Element, those two calls are serialized?

No. It could be atomic in order to ensure the post-condition.

> Why primitive subprograms? What about class-wide subprograms declared in the same package?

That was my question too. Presumably, primitive operations were considered building blocks for class-wide operations, which, under this assumption, would be safe per design for some, rather, weak [as you pointed below] post-conditions.

> Does task safety imply absence of deadlock?

Pragmatically, the answer could be no, if more than one object involved. Yes, for single object.

Safety of any subset of a set of objects is stronger than safety of individual objects.

[...]

From: Brad Moore <brad.moore@shaw.ca>
Date: Wed, 07 May 2014 23:03:21 -0600
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

> [...] the post-condition of the operation [and the object's invariant] is true for any number of tasks Wi calling the operation independently at any point Ti of real-time.

That's a nice definition. It does cover a broad set of cases including pure functions, protected operations, sets of function calls whose parameters do not conflict, mutex guarded calls, etc.

>> [...] What about class-wide subprograms declared in the same package?

As explained in another email, I was worried about non-tagged types. eg.

```
type T is record ... end record with
      Task_Safe => True;
type U is record ... end record with
      Task_Safe => False;

function Bar (X1 : T; X2 : U)
      return Integer;
```

It might be confusing or onerous for the reader to determine if Bar is task safe or not, particularly if there is a long list of parameters.

I think Class-wide subprograms declared in the same package however could probably be lumped in with the primitive functions, and the aspect could apply to those as well..

>> Does task safety imply absence of deadlock?

It would be nice if it could imply the absence of deadlock, but I think that might be too lofty a goal.

> [...]

Even without a formal definition of task safety, the compiler could check that a task safe subprogram only calls other task safe subprograms. (Subprograms that have been explicitly marked by the subprogrammer as being task safe). That'd be a good start actually.

But I think it would be better if the compiler could provide more safety.

A task safe call should probably not be a potentially blocking call, I think.

A task safe subprogram should also not modify global variables that aren't protected, or atomic.

This would add quite a bit of safety, I think but maybe there are other restrictions that could be added also.

From: Brad Moore <brad.moore@shaw.ca>
Date: Thu, 08 May 2014 06:03:38 -0600
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

> [...]

Actually, I have second thoughts about disallowing calls to entries from a "task_safe" subprogram. It should be allowed, as the rules about potentially blocking operations would help prevent calling an entry from another entry.

But I think the idea could be carried further. Another property of a subprogram that is important to know is whether it is a potentially blocking call or not. That is another attribute that would be nice to capture in the contract. I think it would be useful to have a Potentially_Blocking aspect that could be similarly applied to a subprogram specification.

I see it working something like the following;

- A Potentially_Blocking call is viewed conceptually as being a Task Safe call (as it should). It is a more specific kind of a task safe call.
- The Potentially_Blocking aspect may be optionally applied to any subprogram, whether it is potentially blocking or not. If the subprogram is not potentially blocking, it might mean that the programmer is reserving the right to make it a potentially blocking call in the future, or that other implementations of the specification might be potentially blocking.
- If a subprogram directly has task or protected object entry calls, then it cannot be explicitly specified as having the Task_Safe aspect. It must instead be specified as having the Potentially_Blocking aspect. (Alternatively the subprogram can be left without any aspect specification. It is only used if the programmer wants to capture these details in the contract of the subprogram, but then the subprogram is not Task Safe, i.e. the Task_Safe aspect is false)
- A Task Safe program can only call other Task_Safe subprograms or Potentially_Blocking subprograms. If the Task_Safe subprogram calls a Potentially_Blocking Subprogram, then it cannot be explicitly specified as having the Task_Safe aspect. It must instead have the Potentially_Blocking Aspect specified. (or no aspect specification, or specified as false meaning that the subprogram is not Task Safe, i.e. the Task_Safe aspect is false)

Examples:

```
function Foo return Integer with
      Potentially_Blocking;
function Bar return Integer with
      Task_Safe;
```

Bar cannot call Foo, unless the specification is modified to either;

```
function Bar return Integer with
      Potentially_Blocking
```

or

```
function Bar return Integer with
      Task_Safe => False;
```

(or)

```
function Bar return Integer;
```

From: Brad Moore <brad.moore@shaw.ca>
Date: Sat, 10 May 2014 06:30:14 -0600
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

> [...] I prefer old -- -style comments.

It'd be far, far better than a comment, in my mind. A comment doesn't cause compilations to fail. A comment does not improve the safety of a program, only the quality of the code in the sense that uncommented code tends to be harder to read and understand.

While it goes too far to say that the Task_Safe aspect would prove task safety, it would prove that the subprogram does not refer to any unprotected, non-atomic variables in a global scope. It also proves that the subprogram does not call any other subprograms that do the same. That goes a long way on the task safe spectrum

If Foo calls Bar, and both Foo and Bar have the Task_Safe aspect, but some time later the maintainer of Bar decides to change its implementation to refer to some global variable or call some other subprogram that doesn't have the Task_Safe aspect, the compiler would force the programmer to remove the Task_Safe aspect from Bar. This would have a ripple effect, so that a program that calls Foo would fail compilation, and force the maintainer of Foo to remove the Task_Safe aspect on that subprogram. The maintainer of Bar would realize that he is breaking its contract, and might decide to revert his change, or choose a different implementation that allows him to leave Bar's contract intact. With comments, this would have been a maintenance hazard. It's not always obvious to a programmer when such a change is made, that it breaks such assumptions in the client usage of the subprogram. It's also error prone to expect the programmer to exhaustively examine all client usage of that subprogram to check for such assumptions that might have been broken.

The maintainer of Bar might also not be aware that some of the subprograms that Bar calls have dependencies on global variables. The maintainer of Bar should not have to recursively look at the implementation of every subprogram it calls, and every subprogram those subprograms call to see if there are unsafe dependencies on unprotected global variables.

Further, these aspects (Task_Safe, and Potentially_Blocking) would improve the safety of other parts of the standard.

The compiler could be used in a stricter rules checking mode that forbids protected subprograms or entries from calling subprograms that are not Task_Safe, or that are Potentially_Blocking. (At the very least, the compiler could issue warnings.)

Rather than only relying on a run time check to raise an exception when a protected subprogram or entry calls a subprogram that blocks (possibly only in rare circumstances that might be missed during testing), it is much more likely that the problem would have been caught during compile time.

Also, since the compiler cannot prove that calls to other languages such as C are not referring to variables unsafely, the Task_Safe aspect would likely forbid calls to other languages. A Task_Safe subprogram is one written in pure Ada. Some would argue that that alone says a lot about the safety quality of the subprogram.

From: Brad Moore <brad.moore@shaw.ca>
Date: Sun, 11 May 2014 00:56:18 -0600
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

[...]

> Compare it with protected actions. It is safe to call an operation which itself is not protected from a protected operation on the context of a protected action.

But that's only true if the operation is only ever called from within that same instance of the protected object (Something that could be difficult to know without the aspect), and that there is only one instance of that protected type of the protected object. Otherwise it's not safe to call from a protected operation as there could be other concurrent calls calling the unsafe operation.

The following program illustrates this. If you run the program with a small value of N (specified on the command line), say 100, then chances are the program executes correctly to completion. However if you use a larger value of N (say 1_000_000, the default), then the program fails, due to the use of global variables. In Test1, the Unsafe function is called directly from multiple tasks.

In Test2, the same Unsafe function is only called from protected functions, but it still fails.

In both tests, the failures are due to function Unsafe failing its Postcondition.

If the Task_Safe attribute existed, the compiler could have issued a warning at compile time that the tasks were calling subprograms that weren't Task_Safe, and the problems could have been avoided.

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Exceptions; use Ada;
with Ada.Command_Line;

procedure Test_Task_Safety is

  -- Defaults to 1_000_000, but can be
  -- specified on command line
  N : constant Natural := (if
       Command_Line.Argument_Count >= 1
      then Natural'Value
          (Command_Line.Argument (1))
      else 1_000_000);
  Global_Data : Integer := 0;
  function Unsafe (X : Natural)
          return Natural
     with Post => Unsafe'Result = X + 1 --,
      -- Task_Safe => False
     ;
  function Unsafe (X : Natural)
          return Natural is
  begin
    Global_Data := X;
    Global_Data := Global_Data + 1;
    return Global_Data;
  end Unsafe;

begin

  New_Line;
  Put_Line ("********************************* ");
  Put_Line ("****  Test1 : Unsafe calls ***** ");
  Put_Line ("********************************* ");
  New_Line;

  Test1 : declare

    task type T1 is
    end T1;

    task body T1 is
      Result : Natural := 0;
    begin
      for I in 1 .. N loop
        Result := Unsafe (Result);
      end loop;
      Put_Line ("Result =" &
          Natural'Image (Result));
    exception
      when E : others =>
        Put_Line ("Task_Died" &
  Ada.Exceptions.Exception_Information(E));
    end T1;
    Workers : array (1 .. 10) of T1;
  begin
    null;
  end Test1;

  New_Line;
  Put_Line
("************************************************");
  Put_Line ("****  Test2 : Unsafe calls from
                    protected objects ***** ");
  Put_Line
("************************************************");
  New_Line;

  Test2 : declare

    protected PO1 is
      function Foo (X : Natural)
          return Natural;
    end PO1;

    protected PO2 is
      function Bar (X : Natural)
          return Natural;
    end PO2;

    protected body PO1 is
      function Foo (X : Natural)
          return Natural is
      begin
        return Unsafe (X);
      end Foo;
    end PO1;
    protected body PO2 is
      function Bar (X : Natural)
          return Natural is
      begin
        return Unsafe (X);
      end Bar;
    end PO2;

    task type T1 is
    end T1;

    task body T1 is
      Result : Natural := 0;
    begin
      for I in 1 .. N loop
        Result := PO1.Foo(Result);
      end loop;
      Put_Line ("Result =" &
          Natural'Image (Result));
    exception
      when E : others =>
        Put_Line ("Task_Died" &
  Ada.Exceptions.Exception_Information(E));
    end T1;

    task type T2 is
    end T2;

    task body T2 is
      Result : Natural := 0;
    begin
      for I in 1 .. N
        Result := PO2.Bar (Result);
      end loop;
      Put_Line ("Result =" &
          Natural'Image (Result));
    exception
      when E : others =>
        Put_Line ("Task_Died" &
  Ada.Exceptions.Exception_Information(E));
    end T2;

    Foo_Workers : array (1 .. 10) of T1;
    Bar_Workers : array (1 .. 10) of T2;

  begin
    null;
  end Test2;

  null;
end Test_Task_Safety;
```

Output:

```
*********************************
****  Test1 : Unsafe calls *****
*********************************
Task_DiedException name:
SYSTEM.ASSERTIONS.ASSERT_FAILURE
Message: failed postcondition from
test_task_safety.adb:15

[7 identical messages omitted. —sparre]

Task_DiedException name:
SYSTEM.ASSERTIONS.ASSERT_FAILURE
Message: failed postcondition from
test_task_safety.adb:15
Result = 1000000
************************************************
****  Test2 : Unsafe calls from
      protected objects ****
************************************************
Task_DiedException name:
SYSTEM.ASSERTIONS.ASSERT_FAILURE
Message: failed postcondition from
test_task_safety.adb:15

[17 identical messages omitted. —sparre]

Task_DiedException name:
SYSTEM.ASSERTIONS.ASSERT_FAILURE
Message: failed postcondition from
test_task_safety.adb:15

Result = 1000000
```

From: Brad Moore <brad.moore@shaw.ca>
Date: Sun, 11 May 2014 12:01:20 -0600
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

> [...]

A couple more thoughts and refinements to throw in, for consideration.

This got me thinking about what could be done to be able to say with more confidence that a Task_Safe subprogram is in fact task safe (i.e. Safe to call concurrently).

What if we threw in a couple more restrictions.

- A Task_Safe subprogram does not contain any backwards jumping goto statements, nor does it contain while loops. (Or at least while loops that cannot be easily proven to be guaranteed to exit. For loops however are OK, since they are guaranteed to exit.)
- A subprogram that does contain backwards jumping goto statements or while loops are considered to be potentially blocking, for the purpose of the Potentially_Blocking aspect, so applying the Potentially_Blocking aspect to such a subprogram would be allowed.

It would be OK for the main body of a task to have while loops of course. The compiler would just statically warn about calling subprograms that have while loops.

Now it seems to me that when we say Task_Safe, it is more than just documenting the intent of the programmer, it is provable.

Such a subprogram for example could not deadlock because it does not contain any endless loops, and does not call anything that blocks. It also does not refer to any global variables.

With such restrictions, can you provide any example that you would consider unsafe? I am having difficulty coming up with one.

Keep in mind also that Task_Safe is not a term defined in the RM. We can pretty much define it to mean whatever we want, including something that is provable.

From: Brad Moore <brad.moore@shaw.ca>
Date: Tue, 13 May 2014 09:01:44 -0600
Subject: Re: Safety of unprotected concurrent operations on constant objects
Newsgroups: comp.lang.ada

[...]

The goal is to be able to say that a subprogram can be called safely, without erroneousness with other concurrent calls.

Atomicity plays a part of it, but subprograms such as pure functions that don't modify state also fall under the umbrella.

Using discriminated records to return variable amount of data from function

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue Aug 12 2014
Subject: Using discriminated records to return variable amount of data from function
URL: http://ada.tips/using-discriminated-records-to-return-variable-amount-of-data-from-function.html

Sometimes, you want to return different data from a function depending on the given parameters and the program state. For example, when searching a container for a certain element, you either want to return "NOT_FOUND" information or the actual element.

One way to do this is to use discriminated (or variant) records

```
type Search_Result (Found : Boolean) is
record
  case Found is
    when True =>
      Value : Integer;
    when False =>
      null;
  end case;
end record;
```

If 'Found' parameter is True, you also have 'Value' component in the record. And if 'Found' is 'False', you have nothing extra.

The search function itself could be

```
function Search_Numbers (Key : Integer)
        return Search_Result is
begin
  if Key = 99 then
    return Search_Result'
        (Found => True, Value => 101);
  else
    return Search_Result'
        (Found => False);
  end if;
end Search_Numbers;
```

And it can be used like this:

```
procedure Main is
  Res : Search_Result :=
          Search_Numbers (99);
begin
  if Res.Found then
    Put_Line ("Found: " &
        Integer'Image (Res.Value));
  else
    Put_Line ("Not found");
  end if;
end Main;
```

If you try to access 'Value' component of the 'Res' variable when 'Found' is False, an exception is raised during runtime.

I/O Request Queueing for Ravenscar

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Fri, 29 Aug 2014 00:17:12 +0300
Subject: Re: STM32F4 Discovery, communication and libraries
Newsgroups: comp.lang.ada

> [...] You could not implement an equivalent of I/O queueing under the Ravenscar constraints.

It is certainly possible to implement an I/O request queue in Ravenscar; I have done so for the platform SW on ESA's GOCE satellite. Multiple client tasks, one server (interface driver) task. An I/O request contains (or is, or refers to) a client-specific protected object (PO) with an "I/O completed" entry, on which the client task waits after enqueueing the I/O request. The server task processes submitted I/O requests in any order and concurrency it chooses; when an I/O request is done, the server task calls an operation on the request's PO, which unblocks the entry, resuming the client task.

Optimisation

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 11 Sep 2014 15:34:41 +0200
Subject: Re: Assuming optimization? What is best of these code alternatives?
Newsgroups: comp.lang.ada

[...]

FWIW, here is my favorite collection of quotes about optimization:

Kernighan & Plauger, 1974. Elements of Programming Style:

- Make it right before you make it faster.
- Keep it right when you make it faster.
- Make it clear before you make it faster.
- Don't sacrifice clarity for small gains in "efficiency."
- Let your compiler do the simple optimizations.
- Keep it simple to make it faster.
- Don't diddle code to make it faster - find a better algorithm.
- Instrument your programs. Measure before making "efficiency" changes.

Ledgard, Nagin, Hueras, 1979. Pascal with Style: Programming Proverbs:

Shortening the code, running the program faster, or using fewer variables are all popular pastimes. Not mentioning ... the extra testing time needed to check the new and often subtle boundary conditions, are you sure that fewer machine instructions or faster machine execution is likely?

M.A. Jackson, Rules of Optimization:

- Rule 1: Don't do it.
- Rule 2 (for experts only): Don't do it yet.

W.A. Wulf

"More computing sins are committed in the name of efficiency (without necessarily achieving it) than for any other single reason - including blind stupidity."

Donald Knuth

"We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil."

Concurrent Programming Patterns for I/O

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 26 Sep 2014 09:58:44 +0200
Subject: Re: can someone help me with this code (explanation)
Newsgroups: comp.lang.ada

Björn Lundin wrote: [About managing output from multiple tasks. —sparre]

> Or by using a protected object as a semaphore.

But why? It's so simpler with a task:

```
task Printer is
  entry Print (Mess : String);
end Printer;

task body Printer is
  use Text_IO;
begin
  loop
    select
      accept Print (Mess : String) do
        Put_Line (Mess);
      end Print;
    or terminate;
    end select;
  end loop;
end Printer;
```

From: Brad Moore <brad.moore@shaw.ca>
Date: Mon, 29 Sep 2014 22:22:17 -0600
Subject: Re: can someone help me with this code (explanation)
Newsgroups: comp.lang.ada

> Task switches, or tasks in the first place, are, apparently, heavy weight. That's by comparing two Ada programs:

> http://benchmarksgame.alioth.debian.org/u64q/program.php?test=threadring&lang=gnat&id=2

> http://benchmarksgame.alioth.debian.org/u64q/program.php?test=threadring&lang=gnat&id=4

Apparently the heaviness of tasks is dependent on usage.

I just submitted another version that was accepted today, which moves Ada up the ladder a bit.

http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=threadring

In fact, only the Go and the Haskell entries are consistently ahead of this latest version for all 4 processor configurations. The Ada version has 503 tasks executing with calls on a protected entry (where each task maps to an OS thread in GNAT). Looking at the Go example, I'm guessing that the 503 lightweight threads are being executed by a single OS thread, likely under a work-stealing scheduler, where the work never gets stolen by other cores, since this benchmark mostly involves a sequential handoff of a token to the next thread. It that's what's happening, then it might explain why the Go version is still quite a bit faster, since executing the problem on the same core doesn't require as much overhead and locking to do the handoff.

> and then both of them to the "different kind" of parallelism exhibited by the leading entries. The leading entries are faster by an orders of magnitude, even though the faster Ada program uses just semaphores.

>

> It might be better for Ada if at least the parallel loop initiatives announced in Ada Letters are getting somewhere. I'm just guessing at the effectiveness WRT async little things, though.

The proposals are gathering steam. We have moved the design of our proposal along quite a bit since the earlier papers, and even quite a bit since our most recent paper. In fact, our latest paper is being presented at HILT 2014 in Portland next month. The syntax has been revamped and simplified, while providing better information to the compiler so that the compiler may verify if the parallelism can occur while also allowing the compiler to do more implicit parallelism.

[...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 8 Oct 2014 21:45:01 -0500
Subject: Re: can someone help me with this code (explanation)
Newsgroups: comp.lang.ada

> The semaphore solution lets one group related output lines together; the task+entry (in the above form) can interleave output lines from different tasks, perhaps making output harder to read. Whether this matters depends on what one needs.

Right. Note that the semaphore ought to be wrapped in a Limited_Controlled object so that it gets unlocked if the scope is exited by an exception. That's especially important for protecting I/O, since I/O routines have a tendency to propagate an exception because of full disks, permissions errors, etc. Without such protection, an exception would leave the semaphore locked and you'll end up with deadlock as no task can do any I/O. (The task version probably ought to be protected from exceptions as well, I'll leave that as an exercise for the reader.)

Task Allocation?

From: Riccardo Bernardini <framefritti@gmail.com>
Date: Thu, 16 Oct 2014 11:18:40 -0700
Subject: Re: dynamic vs static tasks allocation
Newsgroups: comp.lang.ada

> If you know ahead of time how many tasks will be needed is it better to create tasks dynamically with new operator or just statically?

> Which is way is recommended and why even bother allocating dynamically?

Nice question... I guess that the answer is a definitive "it depends."

Just my 2.0e-2...

As a general rule, I always prefer to avoid dynamic allocation, so I prefer to allocate the task statically. BTW, please note that if you do something like:

```
task type Foo;
declare
  Worker : Foo; -- Worker starts here
begin
  -- Worker is running
  ... do something ...
end;
```

task Worker is started "dynamically" when the execution reaches the "declare" block, without using "new." I do not know if your idea of "dynamically" includes this example or not. Also note that in this case:

```
procedure Bar(N: Positive) is
  Workers : array (1..N) of Foo;
begin
  -- N workers running
  ... do something;
end Bar;
```

The number of tasks is determined dynamically at runtime.

Another reason for having static tasks is (in general) efficiency. The typical example is a web server that waits for connections on the port 80 and every time a connection arrives, it hands the new connection to a sub-server that takes care of all the dialogue with the client. In this case you have two possible macro-approaches: create the sub-server task at runtime with "new" or keep a "pool" of static sub-servers that serve the requests, then go back to sleep, waiting for a new request. I expect the second solution to be more efficient (in terms of time required to reply to the client) since you avoid the overhead related with task creation. (We are on the border of the sin of "preventive optimization" here, a more precise analysis should be done on a case-by-case basis).

Moreover, if I remember correctly, there are some "profiles" that do not allow for the dynamic creation of tasks, so all your tasks must be "static."

Finally, why using "new" for creating new tasks? Well, once I needed to keep a task "inside" a record, but if you declare a component of task type the record will be limited (it does not make any sense to copy a record). Since having the record limited was a problem, I used an access to task and this required to have the task created dynamically. (Sorry, I do not remember the details, it was too much time ago).

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 16 Oct 2014 11:41:51 -0700
Subject: Re: dynamic vs static tasks allocation
Newsgroups: comp.lang.ada

> [...]

> declare

> Worker : Foo; -- Worker starts here

> [...]

> procedure Bar(N: Positive) is

> Workers : array (1..N) of Foo;

> [...]

Also note that in the above examples, the block which declares the task (in the first example) or the procedure Bar (second example) will not be allowed to exit until the task(s) are done (technically, until they are "terminated"). If the block or Bar uses "new" to start the task, the block or Bar can complete while the task is still running, as long as the access type used for "new" is not declared inside the block or Bar (technically, the ultimate ancestor of the access type). That may be another reason to use "new" to create a task.

Open Question: Generic Formals and Aspects?

From: Simon Wright <simon@pushface.org>
Date: Fri, 17 Oct 2014 14:17:35 +0100
Subject: Generic formals and Aspects
Newsgroups: comp.lang.ada

Recently on StackOverflow there was a question[1] about clamping a value to a range.

The answer, so far, suggests a generic:

```
generic
   type Source_Type is range <>;
   type Destination_Type is range <>;
function Saturate (X : Source_Type)
      return Destination_Type;
```

With discussion about what happens if Destination_Type'Range is not a subset of Source_Type'Range. I see in AARM 13.1.1(4.b)[2] that a formal_type_declaration is allowed to include an aspect specification, so tried:

```
generic
   type Source_Type is range <>
     with Static_Predicate =>
       Long_Long_Integer
           (Destination_Type'First)
         >= Long_Long_Integer
           (Source_Type'First)
       and Long_Long_Integer
           (Destination_Type'Last)
         <= Long_Long_Integer
           (Source_Type'Last);
   type Destination_Type is range <>;
   function Saturate (X : Source_Type)
         return Destination_Type;
```

But GNAT (4.9.1, GPL 2014) said that Static_Predicate wasn't allowed (nor was Dynamic_Predicate).

(GNAT specific?) Predicate was allowed, but had no effect: I was able to instantiate with:

```
type Source is new Integer range 10 .. 20;
type Destination is new Integer
      range 30 .. 40;
```

(a) What aspects are/should be allowed in a formal_type_declaration?

(b) How to write the generic to prevent this sort of mistake at compile time? (It is easy enough to get a runtime Constraint_Error.)

[1] http://stackoverflow.com/questions/26390135/can-i-clamp-a-value-into-a-range-in-ada

[2] http://www.ada-auth.org/standards/12aarm/html/AA-13-1-1.html#p4.b

“raise” in Aspects

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Wed, 22 Oct 2014 13:08:09 -0500
Subject: Re: 'raise' in aspects?
Newsgroups: comp.lang.ada

> [...]

For the record:

AI12-0022 adds "raise_expression" to the "relation" syntax. It's a binding interpretation on Ada 2012:

http://www.ada-auth.org/cgi-bin/cvsweb.cgi/ai12s/ai12-0022-1.txt?rev=1.13

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 23 Oct 2014 17:42:08 -0500
Subject: Re: 'raise' in aspects?
Newsgroups: comp.lang.ada

> [...]

Right, "raise_expression" can be used in any expression, not just aspects. Indeed, we immediately noticed that it fixed one long-standing problem in Ada (the need to have a return statement in every function). You can write

```
return raise Program_Error with
      "Not yet implemented";
```

in any function (since a raise expression matches any type), and you don't have to dream up a useless dummy return value to do so.

From: Simon Wright <simon@pushface.org>
Date: Fri, 24 Oct 2014 08:20:13 +0100
Subject: Re: 'raise' in aspects?
Newsgroups: comp.lang.ada

[...]

I liked (but haven't had reason to try; I had already spent far too long generating "useless dummy return values") Bob Duuff's recursive solution:

```
function F return Boolean is
begin
   raise Program_Error with
         "Not yet implemented";
   return F;
end F;
```

Ada-rebirth – The Ada Mascot Competition

From: David Botton <david@botton.com>
Date: Mon, 10 Nov 2014 04:16:04 -0700
Subject: Ada-rebirth – The Ada Mascot Competition
Newsgroups: comp.lang.ada

[...] a single, modern, slick Ada mascot
[...]
Rules:
1. All submissions must be original and you agree are public domain on submission
2. It must be a "being", man, animal, machine or other
3. Ideally have some story to go with the Ada language, but not a requirement to be considered. (Wikipedia Augusta Ada King, Countess of Lovelace and the Ada Language would be places for ideas)
4. It should be "vector art" or something that can be traced (and so pixel dense enough) for vectors later to be used in various media formats.
5. Not a requirement, but line art is always positive or something that can easily be used with many color schemes.
6. Deadline - Feb 14, 2015 - Valentines Day in honor of the Lady Lovelace
7. Winner to be announced on Friday, Feb 27 2015
8. A panel of judges will be selected and announced and submissios will be posted for public comments from Feb 15-27 at http://www.gnoga.com/rebirth.html
(No one that submits an entry will be a judge, nor will I be a judge. All donors that have not submitted entries will be judges and requests are in to Ada Advocacy groups to participate as judges as well.)
9. Send submissions by e-mail to david@botton.com

[see http://www.gnoga.com/#rebirth —sparre]

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

January 04-20 14th **International Conference on Software Reuse** (ICSR'2015), Miami, Florida, USA. Topics include: domain-specific languages; COTS-based development and reuse of open source assets; software product line techniques; generative development, model-driven development; software composition and modularization; software evolution and reuse, and reengineering for reuse; quality assurance for software reuse, such as testing and verification; reuse of non-code artifacts (process, experience, etc.); transition to software reuse and industrial experience with reuse; etc.

January 08-10 16th IEEE **International Symposium on High Assurance Systems Engineering** (HASE'2015), Daytona Beach, 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.

January 13-14 POPL2015 - ACM SIGPLAN **Workshop on Partial Evaluation and Program Manipulation** (PEPM'2015), Mumbai, India. Topics include: program and model manipulation techniques (such as: partial evaluation, slicing, symbolic execution, refactoring, ...); program analysis techniques that are used to drive program/model manipulation (such as: abstract interpretation, termination checking, type systems, ...); techniques that treat programs/models as data objects (including: metaprogramming, generative programming, embedded domain-specific languages, model-driven program generation and transformation, ...); etc. Application of the above techniques including case studies of program manipulation in real-world (industrial, open-source) projects and software development processes, descriptions of robust tools capable of effectively handling realistic applications, benchmarking.

January 19-21 10th **International Conference on High Performance and Embedded Architectures and Compilers** (HiPEAC'2015), Amsterdam, the Netherlands. Topics include: computer architecture, programming models, compilers and operating systems for embedded and general-purpose systems; parallel, multi-core and heterogeneous systems; reliability and real-time support in processors, compilers and run-time systems; architectural and run-time support for programming languages; programming models, frameworks and environments for exploiting parallelism; compiler techniques; etc.

☺ January 21 HiPEAC2015 - 3rd **Workshop on High-performance and Real-time Embedded Systems** (HiRES'2015). Topics include: runtimes and operating systems combining high-performance and predictability requirements; programming models and compiler support for providing real-time capabilities to multi- and many-core architectures; models and tools for code generation, system verification and validation, etc.

January 21-23 9th **International Workshop on Variability Modelling of Software-intensive Systems** (VaMoS'2015), Hildesheim, Germany. Topics include: variability across the software life cycle, separation of concerns and modularity, adaptivity at runtime and development time, programming languages and tool support, case studies and empirical studies, etc. Deadline for early registration: January 5, 2015.

January 27-30 13th **Australasian Symposium on Parallel and Distributed Computing** (AusPDC'2015), Sydney, Australia. Topics include: multicore systems; GPUs and other forms of special purpose processors; middleware and tools; parallel programming models, languages and compilers; runtime systems; reliability, security, privacy and dependability; applications; etc.

♦ January 31 **Ada Developer Room at FOSDEM 2015**, Brussels, Belgium. FOSDEM 2015 is a two-day event (Sat 31 Jan - Sun 01 Feb). This years' edition includes once more a full-day Ada Developer Room, organized by Ada-Belgium in cooperation with Ada-Europe, which will be held on Saturday 31 January.

☺ February 07-11 20th ACM SIGPLAN **Symposium on Principles and Practice of Parallel Programming** (PPoPP'2015), San Francisco Bay Area, USA.

☺ Feb 07-11 PPoPP2015 - **International Workshop on Programming Models and Applications for Multicores and Manycores** (PMAM'2015). Topics include: programming models and systems for multicore, manycore, and clusters of multicore/manycore; multicore and manycore software engineering; automated parallelization and compilation techniques; debugging and performance autotuning tools and techniques for multicore/manycore applications; etc.

February 18-20 8th **India Software Engineering Conference** (ISEC'2015), Bangalore, India. Topics include: software architecture and design, development paradigms, component based software engineering, case studies and industrial experience, software engineering education, static analysis, specification and verification, model driven software engineering, tools and environments, maintenance and evolution, object-oriented analysis and design, distributed software development, etc.

March 02-06 22nd **International Conference on Software Analysis, Evolution, and Reengineering** (SANER'2015), Montréal, Canada. Topics include: all areas of software analysis, evolution, reverse-engineering, and reengineering; empirical studies in reverse engineering; program analysis and slicing; re-documenting legacy systems; reengineering patterns; program transformation and refactoring; mining software repositories for software analysis; software architecture recovery; program comprehension; preprocessing, parsing and fact extraction; reverse engineering tool support; education in reverse engineering; software quality; etc.

March 04-06 7th **International Symposium on Engineering Secure Software and Systems** (ESSoS'2015), Milan, Italy. Topics include: automated techniques for vulnerability discovery and analysis; programming paradigms, models, and domain-specific languages for security; verification techniques for security properties; security by design; static and dynamic code analysis for security; processes for the development of secure software and systems; etc.

March 04-06 23rd **Euromicro International Conference on Parallel, Distributed and Network-Based Computing** (PDP'2015), Turku, Finland. Topics include: embedded parallel and distributed systems, multi- and many-core systems, programming languages and environments, runtime support systems, performance prediction and analysis, shared-memory and message-passing systems, dependability and survivability, real-time distributed applications, etc.

March 09-13 **Design, Automation and Test in Europe Conference** (DATE'2015), Grenoble, France. Topics include: real-time programming languages and software; formal models for real-time systems; worst case execution time analysis; tools and design methods for real-time, networked and dependable systems; dependable systems including safety and criticality; software for safety critical systems; compilers for embedded multi-core, heterogeneous, GPU, reconfigurable, or FPGA platforms; certified compilers; verification techniques for embedded systems ranging from simulation, testing, model-checking, SAT and SMT-based reasoning, compositional analysis and analytical methods; theories, languages and tools supporting model-based design flows covering software, control and physical components; modeling, design, architecture, optimization, and analysis of Cyber-Physical Systems (CPS); case studies in CPS ranging from automotive systems, and avionics, to smart buildings and smart grids; etc.

March 16-19 14th **International Conference on Modularity** (Modularity'2015), Ft. Collins, Colorado, USA. Topics include: varieties of modularity (generative programming, aspect orientation, software product lines, components, ...); programming languages (support for modular abstraction in: language design; verification, specification, and static program analysis; compilation, interpretation, and runtime support; formal languages; ...); software design and engineering (evolution, empirical studies of existing software, testing and verification, composition, methodologies, ...); tools (refactoring; evolution and reverse engineering; support for new language constructs, ...); applications (distributed and concurrent systems; middleware; cyber-physical systems; ...); complex systems; composition; etc. Deadline for submissions: January 12, 2015 (workshop papers).

☺ March 24-27 28th **International Conference on Architecture of Computing Systems** (ARCS'2015), Porto, Portugal. Focus: "reconciling parallelism and predictability in mixed-critical systems". Topics include: models and tools for multi-/many-core systems including but not limited to programming models, runtime systems, middleware, and verification; design, methods, and hardware and software architectures for mixed-critical systems; architectures and design methods/tools for robust, fault-tolerant, real-time embedded systems; etc.

April 11-18 18th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2015), London, UK. Events include: CC (International Conference on Compiler Construction), 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).

> April 12 12th **International Workshop on Formal Engineering approaches to Software Components and Architectures** (FESCA'2015). Topics include: modelling formalisms, temporal properties and their formal verification, interface compliance and contractual use of components, static and dynamic analysis, industrial case studies and experience reports, etc.

April 12-15 23rd **High Performance Computing Symposium** (HPC'2015), Alexandria, VA, USA. Topics include: high performance/large scale application case studies, multicore and many-core computing, distributed computing, tools and environments for coupling parallel codes, high performance software tools, etc.

☺ April 13-17 18th IEEE **International Symposium On Real-Time Computing** (ISORC'2015), Auckland, New Zealand. Topics include: Programming and system engineering (ORC paradigms, languages, model-driven development of high integrity applications, specification, design, verification, validation, testing, maintenance, ...); System software (real-time kernels, middleware support for ORC, extensibility, synchronization, scheduling, fault tolerance, security, ...); Applications (embedded systems (automotive, avionics, consumer electronics, ...), real-time object-oriented simulations, ...); System evaluation (timeliness, worst-case execution time, dependability, end-to-end QoS, fault detection and recovery time. ...); etc. Topics include: object/component/service-oriented real-time distributed computing (ORC) technology; programming and system engineering (ORC paradigms, languages, model-driven development, specification, design, verification, validation, maintenance, time-predictable systems, ...); system software (real-time kernels, middleware support for ORC, extensibility, synchronization, scheduling, fault tolerance, security, ...); applications (embedded systems, real-time object-oriented simulations, ...); system evaluation (timing, dependability, fault detection and recovery time, ...); etc.

April 13-17 30th ACM **Symposium on Applied Computing** (SAC'2015), Salamanca, Spain.

> ☺ April 13-17 **Track on Programming Languages** (PL'2015). Topics include: compiling techniques, domain-specific languages, formal semantics and syntax, garbage collection, language design and implementation, languages for modeling, model-driven development, new programming language ideas and concepts, practical experiences with programming languages, program analysis and verification, programming languages from all paradigms, etc.

> ☺ April 13-17 **Track on Object-Oriented Programming Languages and Systems** (OOPS'2015). Topics include: aspects and components, code generation and optimization, distribution and concurrency, formal verification, integration with other paradigms, software evolution, language design and implementation, modular and generic programming, secure and dependable software, static analysis, testing and debugging, type systems, etc.

> ☺ April 13-17 **Track on Software Engineering** (SE'2015). Topics include: software architecture, and software design patterns; maintenance and reverse engineering; quality assurance; verification, validation, testing, and analysis; formal methods and theories; component-based development and reuse; safety, security, and risk management; dependability and reliability; empirical studies, and industrial best practices; applications and tools; etc.

> April 13-17 **Track on Programming for Separation of Concerns** (PSC'2015). Topics include: software reuse and evolution of legacy systems; consistency, integrity and security; generative approaches; language support for aspect-oriented and SoC systems; etc.

> April 13-17 **Track on Software Verification and Testing** (SVT'2015). Topics include: new results in formal verification and testing, technologies to improve the usability of formal

methods in software engineering, applications of mechanical verification to large scale software, etc.

April 13-17 8th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2015), Graz, Austria. Deadline for submissions: January 16, 2015 (Ph.D. Symposium), February 16, 2015 (Testing Tools track), February 23, 2015 (Testing in Practice papers).

♦ April 20-24 17th **International Real-Time Ada Workshop** (IRTAW'2015), Vermont, New York, USA. In cooperation with AdaCore and Ada-Europe. Deadline for submissions: February 4, 2015 (position papers).

April 22-24 XVIII **Iberoamerican Conference on Software Engineering** (CIbSE'2015), Lima, Peru. Topics include: languages, methods, processes, and tools; reverse engineering and software system modernization; software evolution and maintenance; model-driven engineering; proof, verification, and validation; quality, measurement, and assessment of products and processes; formal methods applied to software engineering; software product families and variability; software reuse; reports on benefits derived from using specific software technologies; quality measurement; experience management; systematic reviews and evidence-based software engineering; industrial experience and case studies; etc.

April 27-29 7th **NASA Formal Methods Symposium** (NFM'2015), Pasadena, California, USA. Topics include: identifying challenges and providing solutions to achieving assurance in mission- and safety-critical systems, model checking, static analysis, modeling and specification formalisms, model-based development, applications of formal methods to aerospace systems and cyber-physical systems, etc.

April 29-30 10th **International Conference on Evaluation of Novel Approaches to Software Engineering** (ENASE'2015), Barcelona, Spain. Topics include: comparing novel approaches with established traditional practices and evaluating them against software quality criteria, software process improvement, model-driven engineering, application integration technologies, software quality management, software change and configuration management, geographically distributed software development environments, formal methods, component-based software engineering and commercial-off-the-shelf (COTS) systems, software and systems development methodologies, etc. Deadline for submissions: January 9, 2015 (position papers).

☺ May 16-24 37th **International Conference on Software Engineering** (ICSE'2015), Firenze, Italy. Topics include: component-based software engineering; debugging, fault localization, and repair; dependability, safety, and reliability; embedded and cyber physical systems; formal methods, verification, and synthesis; middleware, frameworks, and APIs; model-driven engineering; parallel, distributed, and concurrent systems; performance; program analysis; programming, specification, and modeling languages; reverse engineering; security, privacy and trust; software architecture; software economics, management, and metrics; software evolution and maintenance; software modeling and design; software product lines; software reuse; tools and environments; etc. Deadline for submissions: January 13, 2015 (posters), January 23, 2015 (workshop papers, student volunteers), February 15, 2015 (SCORE-it deliverable submission).

May 16-24 **Software Engineering Education and Training** (SEET'2015). Topics include: software and system development; new best practices for SEET; innovative curriculum or course formats; blending software engineering and other engineering disciplines, such as electrical engineering and bioengineering; cooperation in education between industry and academia; continuous education to cope with technological change; etc.

May 16-24 **Track on New Ideas and Emerging Results** (NIER'2015). Topics include: startling results that call into question current research directions, bold arguments on current research directions that may be somehow misguided, etc.

May 25-29 29th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2015), Hyderabad, India. Topics include: parallel and distributed algorithms, applications of parallel and distributed computing, parallel and distributed software, including parallel and multicore programming languages and compilers, runtime systems, parallel programming paradigms, programming environments and tools, etc.

June 13-17 ACM SIGPLAN **Conference on Programming Language Design and Implementation** (PLDI'2015), Portland, Oregon, USA. Topics include: programming language research, including the design, implementation, theory, and efficient use of languages; innovative and creative approaches to compile-time and runtime technology, novel language designs and features, and results from implementations;

	language designs and extensions; static and dynamic analysis of programs; domain-specific languages and tools; type systems and program logics; checking or improving the security or correctness of programs; memory management; parallelism, both implicit and explicit; debugging techniques and tools; etc.
♦ June 22-26	20th **International Conference on Reliable Software Technologies - Ada-Europe'2015**, Madrid, Spain. Sponsored by Ada-Europe, in cooperation with ACM SIGAda, SIGBED, SIGPLAN, and the Ada Resource Association (ARA). Deadline for submissions: January 11, 2015 (papers, tutorials, workshops), January 25, 2015 (industrial presentations).
June 22-26	20th **International Symposium on Formal Methods** (FM'2015), Oslo, Norway. Topics include: interdisciplinary formal methods (techniques, tools and experiences demonstrating formal methods in interdisciplinary frameworks); formal methods in practice (industrial applications of formal methods, experience with introducing formal methods in industry, tool usage reports, etc); tools for formal methods (advances in automated verification and model-checking, integration of tools, environments for formal methods, etc); role of formal methods in software and systems engineering (development processes with formal methods, usage guidelines for formal methods, method integration, qualitative or quantitative improvements); theoretical foundations (all aspects of theory related to specification, verification, refinement, and static and dynamic analysis). Deadline for submissions: January 2, 2015 (abstracts), January 9, 2015 (full papers), February 2, 2015 (industry track papers).
Jun 29 - Jul 01	12th **International Conference on Mathematics of Program Construction** (MPC'2015), Königswinter, Germany. Topics of interest range from algorithmics to support for program construction in programming languages and systems, such as type systems, program analysis and transformation, programming-language semantics, security, etc. Deadline for submissions: January 26, 2015 (abstracts), February 2, 2015 (full papers).
☺ Jun 29 - Jul 02	14th **International Symposium on Parallel and Distributed Computing** (ISPDC'2015), Limassol, Cyprus. Topics include: multi-cores, methods and tools for parallel and distributed programming, tools and environments for parallel program design/analysis, parallel programming paradigms and APIs, distributed software components, parallel embedded systems programming, scheduling, security and dependability, real-time distributed and parallel systems, etc. Deadline for submissions: January 15, 2015 (full papers). Deadline for early registration: May 6, 2015.
July 01-05	39th Annual IEEE **International Computer Software and Applications Conference** (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. Deadline for submissions: January 17, 2015 (papers).
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. Deadline for submissions: January 16, 2015 (workshops).
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. Deadline for submissions: February 1, 2015 (paper abstracts, workshops), February 8, 2015 (papers), March 1, 2015 (tutorials), March 8, 2015 (students events), May 1, 2015 (industrial abstracts).

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. Deadline for submissions: January 30, 2015 (abstracts), February 6, 2015 (papers).
July 20-24	**Software Technologies: Applications and Foundations** (STAF'2015), L'Aquila, Italy.
	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. Deadline for submissions: February 13, 2015 (abstracts), February 20, 2015 (papers).
July 21-23	34th Annual ACM SIGACT-SIGOPS **Symposium on Principles of Distributed Computing** (PODC'2015), Donostia-San Sebastián, Spain.
☺ Aug 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. Deadline for submissions: February 1, 2015 (workshops), March 31, 2015 (papers).
☺ 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. Deadline for submissions: February 28, 2015 (extended abstracts), March 31, 2015 (mini-symposia).
☺ Sep 01-04	44th **Annual International Conference on Parallel Processing** (ICPP'2015), Beijing, China. Topics include: all aspects of parallel and distributed computing.
September 13-16	**Federated Conference on Computer Science and Information Systems** (FedCSIS'2015), Warsaw, Poland.
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. Deadline for submissions: April 12, 2015 (abstracts), April 19, 2015 (full papers).
December 10	200th birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

Preliminary Call for Participation
Ada Developer Room at FOSDEM 2015
31 January 2015, Brussels, Belgium

Organized by Ada-Belgium
in cooperation with Ada-Europe

FOSDEM[1], the Free and Open source Software Developers' European Meeting, is a free and non-commercial two-day weekend event organized early each year in Brussels, Belgium. It is highly developer-oriented and brings together 5000+ participants from all over the world. The 2015 edition takes place on Saturday 31 January and Sunday 1 February. No registration is necessary.

For the 6th time, Ada-Belgium[2] organizes a series of presentations related to Ada and Free or Open Software in a s.c. Developer Room. The “Ada DevRoom” at FOSDEM 2015 is held on the first day of the event, i.e. on Saturday 31 January. The program offers introductory presentations on the Ada programming language, including features of the new Ada 2012 standard, as well as more specialised presentations on focused topics. An important goal is to present exciting Ada technology and projects also to people outside the traditional Ada community. We provide time for discussion and interaction, and organize the by now famous “Adaists dinner” on Saturday evening...

More details are available on the Ada at FOSDEM 2015 web-page, such as the full list with abstracts of presentations, biographies of speakers, and the concrete schedule. For the latest information at any time, contact <Dirk.Craeynest@cs.kuleuven.be>, or see:

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

[1]https://fosdem.org/2015
[2]http://www.cs.kuleuven.be/~dirk/ada-belgium

THE UNIVERSITY of York Ada europe

17th International Real-Time Ada Workshop - IRTAW 2015

in cooperation with AdaCore and Ada-Europe

www.cs.york.ac.uk/~andy/IRTAW2015

Vermont, USA

Week of 20-24 April 2015 (actual dates TBD)

Call for Papers

Since the late Eighties the International Real-Time Ada Workshop series has provided a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions, and has attracted participation from key members of the research, user, and implementer communities worldwide. Recent IRTAW meetings have significantly contributed to the Ada 2005 and Ada 2012 standards, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar profile.

In keeping with this tradition, the goals of IRTAW-17 will be to:

- review the current status of the Ada 2012 Issues that are related with the support of real-time systems;
- examine experiences in using Ada for the development of real-time systems and applications, especially – but not exclusively – those using concrete implementation of the new Ada 2012 real-time features;
- report on or illustrate implementation approaches for the real-time features of Ada 2012;
- consider the added value of developing other real-time Ada profiles in addition to the Ravenscar profile;
- examine the implications to Ada of the growing use of multiprocessors in the development of real-time systems, particularly with regard to predictability, robustness, and other extra-functional concerns;
- examine and develop paradigms for using Ada for real-time distributed systems, with special emphasis on robustness as well as hard, flexible and application-defined scheduling;
- consider the definition of specific patterns and libraries for real-time systems development in Ada;
- identify how Ada relates to the certification of safety-critical and/or security-critical real-time systems;

- examine the status of the Real-Time Specification for Java and other languages for real-time systems development, and consider user experience with current implementations and with issues of interoperability with Ada in embedded real-time systems;
- consider the lessons learned from industrial experience with Ada and the Ravenscar Profile in actual real-time projects;
- consider the language vulnerabilities of the Ravenscar and full language definitions;
- consider testing for compliance with the Real-Time Annex.

Participation at IRTAW-17 is by invitation following the submission of a position paper addressing one or more of the above topics or related real-time Ada issues. Alternatively, anyone wishing to receive an invitation, but for one reason or another is unable to produce a position paper, may send in a one-page position statement indicating their interests. Priority will, however, be given to those submitting papers.

Submission Requirements

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

Authors with a relevant paper under consideration at Ada-Europe (deadline 11th January, 2015) may offer an extended abstract of the same material to IRTAW-17.

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

Important Dates

- Paper Submission: **4 February, 2015**
- Notification of Acceptance: 1 March, 2015
- Confirmation of Attendance: 14 March, 2015
- Final Paper Due: 1 April, 2015
- Workshop: April TBD in week of 20-24, 2015

Program Chair

- Andy Wellings, University of York

Workshop Chair

- Robert Dewar, AdaCore

Program Committee Members

Mario Aldea Rivas, John Barnes, Ben Brosgol, Alan Burns, Michael Gonzàlez Harbour, José Javier Gutiérrez, Stephen Michell, Brad Moore, Luís Miguel Pinho, Juan Antonio de la Puente, Jorge Real, Jose F. Ruiz, Joyce Tokar, Tullio Vardanega, Andy Wellings and Rod White.

Call for Papers

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

22-26 June 2015, Madrid, Spain

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

Ada-Europe 2015

Conference Chair

Alejandro Alonso
Universidad Politécnica de Madrid
alonso@dit.upm.es

Program co-Chairs

Juan A. de la Puente
Universidad Politécnica de Madrid
jpuente@dit.upm.es

Tullio Vardanega
Università di Padova
tullio.vardanega@unipd.it

Tutorial Chair

Jorge Real
Universitat Politècnica de València
jorge@disca.upv.es

Exhibition Chair

Santiago Urueña
GMV, Spain
suruena@gmv.com

Industrial co-Chairs

Jørgen Bundgaard
Rambøll Danmark A/S
jogb@ramboll.dk

Ana Rodríguez
Silver Atena Spain
ana.rodriguez@silver-atena.es

Publicity Chair

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

Local Chair

Juan Zamorano
Universidad Politécnica de Madrid
jzamora@fi.upm.es

ETSIT UPM

"In cooperation" with ACM SIGAda, SIGBED, SIGPLAN, and with ARA

acm In-Cooperation

General Information

The 20th International Conference on Reliable Software Technologies – Ada-Europe 2015 will take place in Madrid, Spain. 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

11 January 2015	Submission of regular papers, tutorial and workshop proposals
25 January 2015	Submission of industrial presentation proposals
1 March 2015	Notification of acceptance to all authors
29 March 2015	Camera-ready version of regular papers required
12 April 2015	Industrial presentations abstracts required
17 May 2015	Tutorial and workshop materials 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.

Topics of interest to this edition of the conference include but are not limited to:

- **Multicore and Manycore Programming**: Predictable Programming Approaches for Multicore and Manycore Systems, Parallel Programming Models, Scheduling Analysis Techniques.
- **Real-Time and Embedded Systems**: Real-Time Scheduling, Design Methods and Techniques, Architecture Modelling, HW/SW Co-Design, Reliability and Performance Analysis.
- **Mixed-Criticality Systems**: Scheduling methods, Mixed-Criticality Architectures, Design Methods, Analysis Methods.
- **Theory and Practice of High-Integrity Systems**: Medium to Large-Scale Distribution, Fault Tolerance, Security, Reliability, Trust and Safety, Languages Vulnerabilities.
- **Software Architectures**: Design Patterns, Frameworks, Architecture-Centred Development, Component-based Design and Development.
- **Methods and Techniques for Software Development and Maintenance**: Requirements Engineering, Model-driven Architecture and Engineering, Formal Methods, Re-engineering and Reverse Engineering, Reuse, Software Management Issues, Compilers, Libraries, Support Tools.
- **Software Quality:** Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems.
- **Mainstream and Emerging Applications**: Manufacturing, Robotics, Avionics, Space, Health Care, Transportation, Cloud Environments, Smart Energy systems, Serious Games, etc.
- **Experience Reports in Reliable System Development**: Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics.
- **Experiences with Ada and its Future**: Reviews of the Ada 2012 new language features, implementation and use issues, positioning in the market and in the software engineering curriculum, lessons learned on Ada Education and Training Activities with bearing on any of the conference topics.

Call for Regular Papers

Authors of regular papers which are to undergo peer review for acceptance are invited to submit original contributions. Paper submissions shall not exceed 14 LNCS-style pages in length. Authors shall submit their work via EasyChair following the link https://easychair.org/conferences/?conf=adaeurope2015 on the conference web site. 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 start of the conference. The authors of accepted regular papers shall prepare camera-ready submissions in full conformance with the LNCS style, not exceeding 14 pages and strictly by March 29, 2015. 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 CiteSeerX Venue Impact Factor has the Conference in the top quarter. Microsoft Academic Search has it in the top third for conferences on programming languages by number of citations in the last 10 years. 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 January 25, 2015. Submissions shall be made via EasyChair following the link https://easychair.org/conferences/?conf=adaeurope2015. 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 April 12, 2015, 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/) host the proceedings of the Industrial Program of the Conference. For any further information please contact the Industrial Chair directly.

Call for Tutorials

Tutorials should address subjects that fall within the scope of the conference and may be proposed as either half- or full-day events. 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 Conference 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.

Tools to get you there. Safely.
Ada and The GNAT Pro High-Integrity Family
www.adacore.com
AdaCore
The GNAT Pro Company

AdDoc (beyond a document generator)

Robert CHOLAY
email: robert.cholay@systerel.fr

Abstract

Code without documentation is generally useless. Moreover, all safety standards require that documentation to be consistent with the actual code. AdDoc automates the generation of documentation from the source code and therefore helps ensuring that the documentation is always up to date.

Keywords: ASIS, EN50128:2011, document generator.

1 Introduction

For the development of its new generation of Railway Control System, Alstom Transportation has decided to improve the features of the tools involved in the development process of safety critical applications. The produced software must conform at least with the EN50128:2011 standard (this includes the produced software and its documentation).

The concerned application is ~450 KLOC Ada2005 with a Software Safety Integrity Level of 4 (the highest).

For this application, Alstom has decided to use a tool able to produce the design documentation in an automatic manner from Ada source files.

2 Method

To avoid wasting energy or resources in "reinventing the wheel", a study of existing tools has been done to find which one could "make the job" with a minimum of effort. Even if there are already very efficient tools, the effort to adapt them to the specific features wanted by Alstom (see below) was judged too important. Thus, Alstom has decided to develop its own tool named AdDoc (Ada Document generator). This study also convinced Alstom that the only reliable implementation solution should be based on the ASIS technology. But why use such a powerful technology as ASIS for documentation purpose only? It was therefore also decided to add some new features to the tool in order to improve the quality of the documentation and of the source code.

3 Tool's behaviours

The current AdDoc tool:

- Checks the completeness of comments and their consistency with the code (based on dedicated rules: all units and subprograms must have an overall description comment and a design description comment, all sort of formal parameters, variables and types must be described throughout a dedicated comment, the structure of composite types must also be documented in a transitive way,…) → this behaviour forces the developer to comment his code,
- Checks some coding rules in order to emphasize source readability (no default in mode parameters, no identifiers factorisation, specification required for subprograms, …) → this behaviour improves source readability,
- Gives the possibility (through configuration files and flags) to:
  - define rules to map a set of Ada units with modules (the tool generates one PDF document per module),
  - ignore a set of compilations units or directories,
  - ignore unit bodies,
  - use the tool with a cross compiler,
  - raise warnings regarding the use of certain predefined Tags,

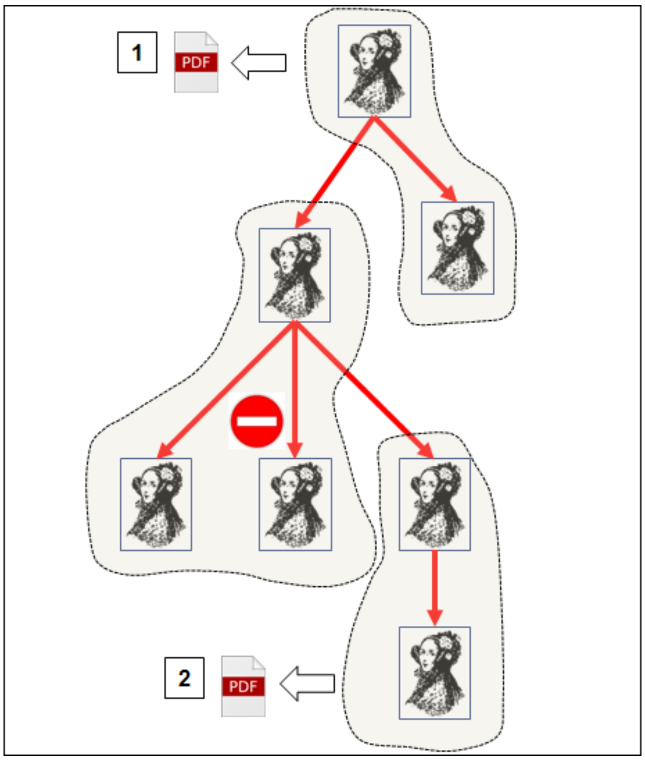


Figure 1 Module configuration

4 Link with the EN50128:2011

As required by the EN50128:2011[3] standard, the design documentation shall address:

1. Identification of all lowest level software units (e.g. subroutines, methods, procedures) traced back to the upper level,
2. Their detailed interfaces with the environment and other components with detailed inputs and outputs,
3. Their safety integrity level without any further apportionment within the component itself,
4. Detailed algorithms and data structures.

The documentation produced by AdDoc has been considered to satisfy points 1), 2), and 3). Regarding the last one, the expected information is available in a document that was already present in Alstom's referential. AdDoc is a tool categorized into the class T1: "generates no outputs which can directly or indirectly contribute to the executable code (including data) of the Software".

5 Under the hood

AdDoc is an Ada2005 ASIS application of ~5KLOC that has been specified, developed and validated in approximately two man-months.

Specifying and developing this kind of application in such a short time implied to make some "short cuts" using specific GNAT [1] implementation facilities and also some ASIS extensions [2].

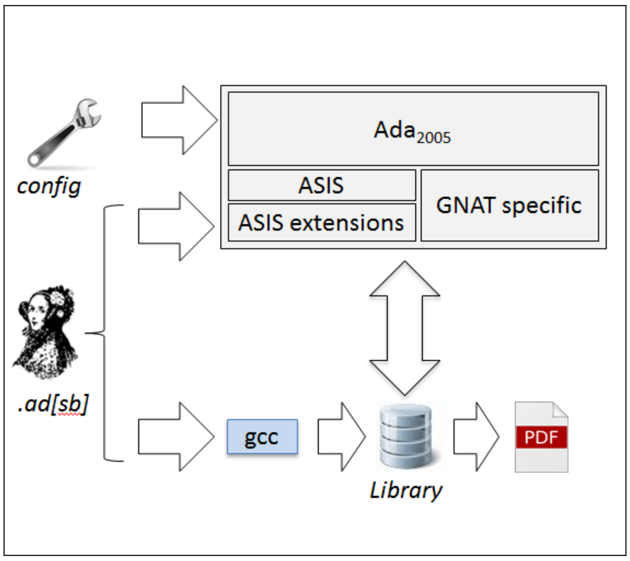


Figure 2 Under AdDoc's hood

One of the difficulties was to access the comment of a syntactic element.

Therefore, the root package AdDoc.Comments has been developed to provide this functionality in a convenient way.

```
package AdDoc.Comments is
   type Comments_T is tagged private;
   No_Comments : constant Comments_T;
   function Read(Element : in Asis.Element) return
   Comments_T;
   ...
private
   ... -- the private part
end AdDoc.Comments;
```

The figure below presents two cases of comments extraction within a syntactic structure which is here a record with a discriminant.

AdDoc is able to treat all kinds of Ada 2005 Asis.Element if a rule has been explicitly defined regarding the way of using comments and the information they should provide.

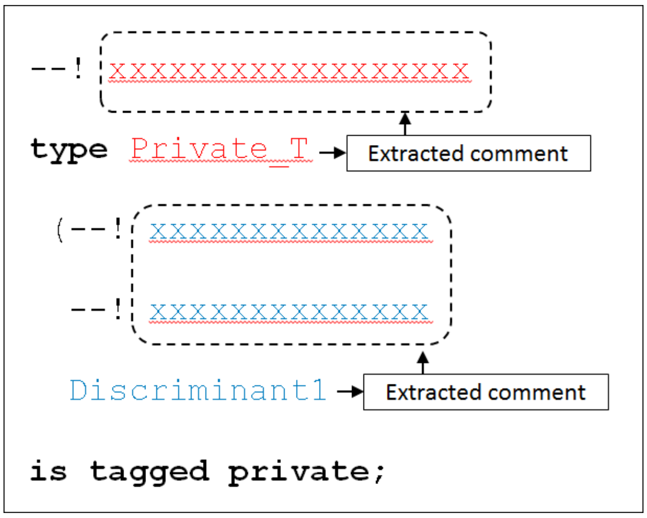


Figure 3 Comments extraction

6 Brief example

As explained above, the document generation process guarantees:

- Consistency of source code formatting,
- Completeness and consistency of comments.

For example the following code is not correct for AdDoc because:

- Param1 and Param2 are not commented (AdDoc found Param and Param3 which does not exist).
- Param3 is not commented (AdDoc found Param which does not exist).
- The default *in* mode of Param2 is missing.

```
--! generic sub-program description.
--! @gen_param  Param description of Param1
--! @gen_param  Param3 description of Param2
--! @gen_param  Proc description of Proc
--! @gen_param  Func description of Func
--! @gen_param  Pkg description of Pkg
--! @param      Input1 description of Input1
--! @param      Input2 description of Input2
--! @return     return function description
generic
type Param1 is (<>);
Param2 : Integer;
with procedure Proc (A : in Integer);
with function Func (B : in Integer) return Boolean;
with package Pkg is new Ada.Text_Io.Integer_Io(<>);
function Generic_Image (Input1: in Integer;
                        Input2 : in Integer) return Integer;
```

If no error is detected, AdDoc generates one or several PDF documents with the expected formatting.

7 Future tool improvements

AdDoc is a tool working on both Windows and Linux with some improvements already planned such as:

- Having a full requirements tag extraction in order to capture and to follow them throughout the source code,
- Providing a more convenient way to define document templates,
- Generating indexes and cross reference tables,
- Having a full integration into GPS (+ documentation),
- Minimising (or removing?) implementation dependences,
- Using Ada2012 facilities for the implementation,
- Supporting Ada2012 constructs,
- Having more coding rules and providing a more convenient way to add them,
- Adding some specific tags for a better LaTeX output (bold, italic, underline,...).

Results and conclusion

AdDoc has now been successfully deployed and integrated in the development process of safety critical application projects.

To be efficient, it should be used at the beginning of the development phase.

Contrary to what was expected, the difficulty was not ASIS but it was the specification of AdDoc:

- Define the rules to apply to comments,
- Identify all the possible syntactic constructions and the comments to apply on.

AdDoc is another example that the strength of the Ada language is not only based on its own quality but also on its ability to promote efficient and powerful technology like ASIS.

References

[1] AdaCore, http://www.adacore.com/

[2] ASIS, http://docs.adacore.com/asis-docs/asis_rm.htm.

[3] Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems; BS EN 50128:2011.

Proceedings

Workshop

Challenges and New Approaches for Dependable and Cyber-Physical System Engineering (De-CPS 2014)

Ada-Europe 2014

23 June 2014

Paris, France

Program

Safety Session

"Three Theses for Complex Model Engineering"
Antoine B. Rauzy (Chaire Chair Blériot-Fabre - Centrale-Supélec, Safran)

"The Use of Controlled Vocabularies and Structured Expressions in the Assurance of CPS"
Katrina Attwood, Philippa Conmy and Tim Kelly (Rapita System and University of York)

Industrial Session

"Dependable Real-Time System and Mixed-Criticality: Seeking Safety, Flexibility and Efficiency with KRON-OS"
Vincent David, Adrien Barbot and Damien Chabrol (Krono-Safe)

"Behavioral Contracts for Energy Consumption"
Shin Nakajima and Masumi Toyoshima (National Institute of Informatics, Tokyo, and DENSO)

"Feasibility Study in the use of contract-based approaches to deal with safety-related properties in CPS"
Daniela Cancila, Elie Soubiran and Roberto Passerone (CEA, Alstom and University of Trento)

Invited Speaker

Charles Robinson (Thales) project manager of ITEA MERGE safety & security project

IMDEA Session

"Formal Verification in Model-based Design of Cyber-Physical Systems"
Pavithra Prabhakar (IMDEA)

Closing Session

"The OMG standard MARTE : feedback and industrialization"
Laurent Rioux (Thales)

Roundtable

Organizing and Program Committee

Organizers: Daniela Cancila, Laurent Rioux

Steering Committee: Antoine B. Rauzy

Program Committee: Katrina Attwood, Benoit Caillaud, Philippa Conmy, Vincent David, Huascar Espinoza, Ali Koudri, Pavithra Prabhakar, Roberto Passerone, Alejandra Ruiz, Bran Selic, Safouan Taha, Masumi Toyoshima.

Publicity Chair: Karima Nahhal, Jean-Louis Gerstenmayer

Challenges and New Approaches for Dependable and Cyber-Physical System Engineering (De-CPS), Ada-Europe 2014

Daniela Cancila, Jean-Louis Gerstenmayer
CEA, LIST, CEA Saclay - F91191 Gif-sur-Yvette Cedex; email: {firstname.name}@cea.fr

The scientific view underlying De-CPS

In June 2014, we have organized in the ECE campus [1] (Paris) the workshop 'Challenges and new Approaches for dependable and Cyber-Physical Systems engineering' (De-CPS) [2], as satellite event of the 19th International Conference on Reliable Software Technologies – Ada-Europe 2014.

In recent years, we have witness a crescendo of industrial and research interest in Cyber-Physical Systems (CPS). One distinguishing trait of CPS is that they integrate software control and decision making with signals from and sensing of an uncertain and dynamic environment. CPS often involve heterogeneous systems, and their design makes extensive use of (tools, systems, languages) interfaces and models. The Horizon 2020 program framework of the European Union devote considerable attention to various aspects of the CPS challenges. A similar trend exists in the EIT ICT Labs [3], a Knowledge and Innovation Communities set up by the European Institute of Innovation and Technology to drive European leadership in ICT innovation for economic growth and quality of life.

The inherent complex and heterogeneous nature of Cyber-Physical Systems impacts the usual methodologies and techniques for critical and real-time embedded systems conception (multiplied interfaces, massive connectivity, dynamical aspects, mix-critical paths of information/causality chain..).

Thereafter, these coming issues were constituting the cornerstone of the workshop :

- how to increase guarantees, according to the dependability objectives for the CPS ;
- how to maintain a high level of control, according to the expected real-time and performance properties ;
- how to better perform, regarding their intrinsic mix-criticality; and finally
- how are evolving the energy consumption issues.

Without the intend to settle the mentioned arguments, the workshop gathers industrial practitioners and research actors to address dependability and, more in general, critical features in CPS.

[1]Ecole d'Ingénieurs http://www.ece.fr/
[2]http://www.ada-europe2014.org/De-CPS.html
[3]http://www.eitictlabs.eu/about-us/strategy/

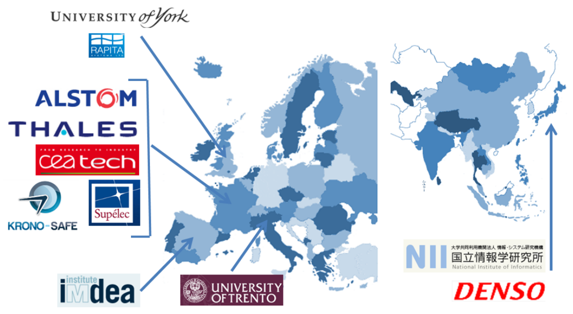


Figure 1: Program

The De-CPS organization has deliberately left the freedom to the authors to publish their contribution in this special issue or to publish their presentation on the ECE web site.

The total number of attendees to the workshop has been fourteen, in equal part from academics and industries. Most of them come from European, by exception of a Japanese delegation (National Institute of Informatics and DENSO). The eight contributions to De-CPS come from:

- three large industrial groups (Thales, Alstom and DENSO);
- two SME (Krono-Safe and Rapita System);
- six centers of research and academics (Supelec, CEA, IMDEA, university of Trento, University of York, National Institute of Informatics).

Figure 1 is a screen-shot of the companies and research centers, which have contributed to the success of the workshop.

Sponsor

The workshop has been partially founded by Krono-Safe, a French and a promising young SME that work on real-time.

ECE has provided the locals for the workshop and Ada-Europe the overall organization.

The Use of Controlled Vocabularies and Structured Expressions in the Assurance of CPS

***Katrina Attwood*[1]*, Philippa Conmy*[2] *and Tim Kelly*[1]**

[1] *Department of Computer Science, University of York, Deramore Lane, YORK YO10 5GH, United Kingdom; Tel +44 1904 325460; email: {katrina.attwood/tim.kelly}@york.ac.uk*

[2] *Rapita Systems Ltd, Atlas House, Osbaldwick Link Road, YORK YO10 3JB, United Kingdom; Tel +44 1904 413945; email: pconmy@rapitasystems.com*

Abstract

To date, work on the development of assurance cases has largely been concerned with the broad structure and content of arguments to contextualise the data. However, at a more detailed level, use of natural language in an argument can lead to conflicting terminology, to difficulties in understanding the nature of the claims being made or to logical inferences which are obscure to the readers of the argument. This problem has become increasingly complex as more and more suppliers are involved in the development chain, making it more difficult to evaluate the strengths and weaknesses of assurance data or to re-use it. This paper explores the development of controlled vocabulary and structured expressions for CPS in the automotive domain, using the Semantics of Business Vocabulary and Business Rules (SBVR) to improve communication and to provide presents some formal consistency checking of content. We highlight the challenges this work has exposed.

Keywords: safety, assurance, controlled language, SBVR, automotive.

1 Introduction

The presentation of assurance cases is now standard practice in a number of safety-critical domains and is mandatory in several. Assurance cases typically comprise both reasoned arguments justifying claims relating to the safety, integrity and/or dependability of CPS and a variety of supporting evidence – analysis and test data, design information and process documentation. Although a considerable body of literature regarding safety-case praxis has been produced, the primary focus to date has been to provide guidance on the structure and content of the arguments, with relatively little attention paid to the language used to convey them. Graphical notations developed for the safety assurance domains (for example, the Goal Structuring Notation (GSN) [1] and the Claims-Argument-Evidence method [2]) inevitably foreground – and simplify – issues of logical flow and the overall readability of the argument, but provide limited guidance on how assertions and supporting statements should be phrased to ensure that the argument is correctly conveyed to a reader or assessor. In the GSN Community Standard [1], for example, less than 10% of the document is devoted to language issues as opposed to the definition, graphical representation, construction and review of argument structures. In practice, many assurance cases are not documented using graphical notations, but use either natural language alone or a combination of natural language and graphical notation for summary purposes.

Imprecise phrasing in assurance cases can lead to a number of problems, including:

- ***Inconsistency*** – terms may be used with different meanings at different points across an argument. This may lead to uncertainties in interpretation, particularly in the subjects of claims and assertions and the scope within which they are valid.
- ***Vagueness*** – without a precise definition of terminology, the author's intended meaning may not be properly conveyed to the audience, whether because there is no shared understanding of the terms used or because there is a failure to 'pin things down' adequately.
- ***Lack of focus in claims*** – in freeform text, it can be difficult to 'unravel' sentence structure so as to establish the scope of terms, i.e. how they influence other terms beyond the single phrasal structure in which they occur [3]. It can therefore be difficult to identify the claims the argument is making, since the relationships between the elements under discussion may not be made clear.

CPS are increasingly assembled by integrator organisations, using multiple components from a diffuse, multinational supply-chain [4]. Compositional approaches to certification mean that assurance data relating to discrete components need to be collected and matched to form an integrated system argument. There is a clear need for consistent usage of domain- and system-specific terminology throughout the supply-chain, and for a shared understanding of the nature and limitations of the claims and evidence being presented in the argument, and of the assumptions made about the operational context in which component behaviour is guaranteed.

We believe there is scope to use controlled language to provide more rigorous rhetorical structure in assurance cases for CPS. We propose a dual approach to address the problems of inconsistency and imprecision outlined above. First, we address semantic aspects by developing a domain dictionary, which provides unambiguous definitions of relevant concepts in the domain over which the argument ranges. Secondly, syntactic aspects are addressed by these definitions to specify claim types in the form of structured expressions to clarify the argument logic. The OMG's Semantics of Business Vocabulary and Business Rules (SBVR) specification [5] offers one means to implement this approach. SBVR provides for the formalized definition of domain concepts, together with the rules and assumptions which define the relationships between them. It contains an explicit model of formal logic, and thus provides a means for the capture of natural language expressions in a formal structure, suitable for machine-processing.

Two of the elements defined in SBVR are of particular significance for our approach: 'concepts' and 'fact types'. These form the basis for the development of the controlled lexicon and claim typology described in the two following sections.

2 Argument semantics: development of a controlled lexicon for safety assurance

In SBVR, a 'concept' is defined as "a unit of knowledge created by a unique combination of characteristics" [5]. Generally, this equates to a noun, or a noun-phrase (also referred to as a 'term'). In SBVR, concepts can be defined formally or informally. In a formal definition, each of the concepts referred to must be defined elsewhere in the vocabulary, thus making for a closed lexicon. Reserved terms to represent logical relationships between concepts are defined in [5]. The "General Concept" and "Concept Type" attributes can be used to specify hierarchical type-relationships between concepts. This is especially useful in the disambiguation of terminological mismatches in cross-domain "translation" scenarios, such as the comparison of concepts across different safety standards.

Our work in the OPENCOSS project [6] defined a preliminary SBVR vocabulary of concepts for assurance arguments. As in the SBVR specification [5], a graphical summary of concept relationships is provided for ease of reference (for human readers). The vocabulary provides a controlled language definition of concepts, artefacts and processes used in the domains of interest of OPENCOSS (railway, avionics and automotive), and thus provides a basis for comparison of usage between the domains. We do not seek to develop a unified, universal lexicon for assurance to be used across the target domains. Such an enterprise is fraught with difficulty, since the certification approaches differ fundamentally. As an illustration, consider the difficulties for a manufacturer seeking to reuse software developed according to IEC 61508 [7] in an avionics context, where certification to DO-178B is required [8]. An assurance argument in the original context – here expressed using SBVR, for clarity – might assert that "software module Y is developed to safety integrity level SIL 4". In the avionics context, the manufacturer may wish to make a similar claim: "software component Y is developed to design assurance level DAL A". Since both the safety integrity level and the design assurance level are instantiations of the generic SBVR concept "Criticality Level" defined by OPENCOSS, it might be assumed that a direct 'translation' between the claims is possible. Examination of the diagrams summarizing the concept relationships for system and software architectures extracted from the SBVR vocabulary we have developed for the relevant standards, however, reveals that the situation is more complicated.

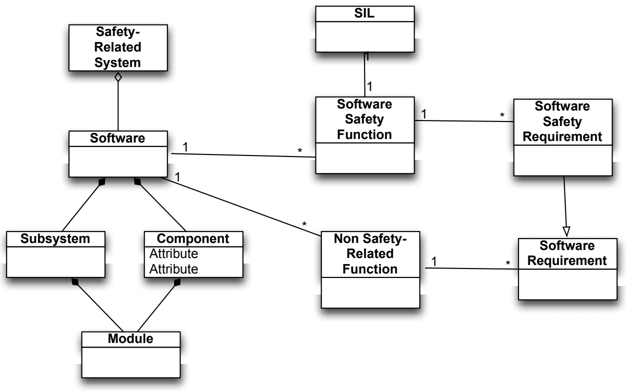


Figure 1 IEC 61508 software concept relationships

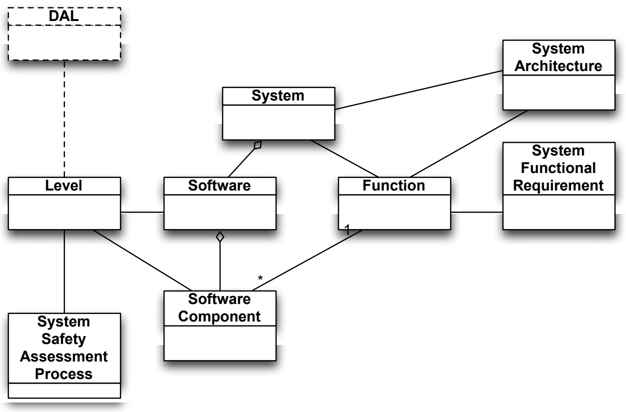


Figure 2 DO-178B software concept relationships

In IEC 61508, a SIL is directly attached to a (software) safety function which is modelled at system level. In DO-178B, however, the DAL is associated with a software system or component, and does not address the "function" concept at all. This implies that direct 'translation' of the claim cannot be made – it is not possible to convert a SIL directly into a DAL without considering the extra process-related concepts that arise because of the focus in DO-178B on the design of the system, rather than merely its functionality. Although a clear understanding of the terminology can be helpful in addressing this difficulty, what is required is not a definition of individual concepts in isolation, but an appreciation of the interrelationships between the concepts, since these provide constraints on reuse of the claim – and associated assurance data – here.

Use of a closed SBVR vocabulary will ensure that these interrelationships are correctly identified. We should therefore consider there to be either a "partial map" or a "no map" relation between the concepts, and a full explanation of the discrepancies between the conceptual structure of the standards is required in order for an engineer to make informed decisions about the feasibility of or limitations on reuse, and on what extra assurance data may need to be provided in the DO-178B context.

A primary concern for the OPENCOSS project is to support reasoning about whether certification artefacts, such as analysis results, can be reused across domains and from one development project to another. In order to support this, the OPENCOSS vocabulary defines terminology at three levels of abstraction: we define vocabulary models to capture the generic vocabulary of safety standards relevant to the domains, organisation-specific terminology and project-specific terminology. Mapping relationships between concepts are used to capture traceability relationships between generic and system-specific concepts (e.g. the fact that a project-specific test plan is an instance of the test plan defined in the organisational model) and also to indicate the degree of "mapping" between concepts at the various levels (e.g. the degree to which the organisational definition of a test plan matches the characteristics of the generic artefact defined in the standard model and relating to a requirement of the standard).

The demonstration of assurance is a much wider and more complex concern than simply establishing conformance to a standard; and an argument is much more than a compliance checklist of processes and artefacts. Having clear definitions of terminology in which concepts are related both vertically by type and sub-type relations and horizontally by being defined in terms of one another in a closed lexicon can help in ensuring consistency of reference across assurance case modules. In particular, the terminology can be used to characterise the interfaces and interdependencies between argument modules, and to ensure that the terms of reference here are consistently understood. The layered vocabulary defined for the OPENCOSS project allows us to clarify the relationships between standards, industrial praxis and development projects, using the "mapping" relationships between concepts at the various levels of abstraction to make any gaps between standards' requirements and projects' actualities clear.

3 Argument semantics: structured claim types

One important means of maintaining consistency in the natural language used to convey the reasoning in an assurance argument is to specify types of claims. A taxonomy of claims can be superimposed on the general concerns of an argument structure identified in the literature (e.g. [9]) and can then be used to refine the logical structures provided in the argument fragment templates captured in GSN patterns such as those presented in [10]. The claim types characterise the types of concepts which are discussed in a particular part of the argument, and the features which are asserted in claims. We have identified several generic claim types for assurance arguments, as summarised in Table 1:

Claim Type	Definition
Activity-Artefact Claim	Claim relating to the production of particular artefacts as a result of particular safety analysis or development activities.
Artefact Compliance Claim	Claim relating to the presentation of a particular artefact necessary for compliance.
Artefact Adequacy Claim	Claim relating to the adequacy and appropriateness of a particular artefact, i.e. moving beyond compliance to a justification of the evidence artefacts provided. E.g., the adequacy of a fault tree
Activity Compliance Claim	Claim relating to the presence and features of features of a safety analysis or development activity necessary for compliance
Activity Adequcy Claim	Claim relating to the adequacy and appropriateness of a particular safety analysis or development activity
Component Development Claim	Claim relating to the adequacy and acceptability of the process by which a component has been developed
Fault Accommodation Claim	Claim relating to the accommodation or elimination of a fault
Hazard Mitigation Claim	Claim relating to the adequacy of hazard mitigation achieved by safety measures in the design

Table 1: Generic claim types for assurance

We can exploit the layered structure of the OPENCOSS vocabulary – where concepts are defined and "mapped" at the level of the standard, the industry model and the project – by defining domain-specific versions of these claim types in parameterised phrases used to populate the GSN argument patterns. These phrases can then be instantiated in component- or system-specific arguments using vocabulary relevant to that component derived from the project vocabulary model. The "Concept Type" mechanism in SBVR allows for the presentation of a series of potential instantiations of a given parameter from which the user can choose. In some cases, the "fact Type" mechanism in SBVR allows to generate the domain-specific claim type directly from the standard or industry vocabulary model.

The "Fact Type" in SBVR [5] is used to capture relationships between concepts defined in the vocabulary. A fact type is defined in [5] as "the meaning of a verb phrase that involves one or more nouns, whose instances are all actualities". A fact type thus equates to a proposition ranging over the concepts represented by the nouns or noun-phrases, a statement of some relationship which can be evaluated logically as having a truth value. As with concepts, fact types can be defined formally – by means of a closed expression in which every term is defined elsewhere in the SBVR model – or informally, using terminology which is not controlled.

In some cases, the "fact type" mechanism in SBVR allows us to generate the domain-specific claim type, and the mapping between the standard (or industry) vocabulary and the project vocabulary provides possible terms with which the template phrase can be instantiated. For claims of the Activity-Artefact type, for example, the SBVR vocabulary derived from the terminology used in the safety standard should identify the types of concept over which the claim might range, by identifying relationships between particular activities and the artefacts they generate. A generic fact type of the sort artefact is generated by activity, for example, can be instantiated by traversing the SBVR "Concept Type" and "General Concept" fields in the standard-level vocabulary to identify a series of individual concepts of type "artefact" and type "activity"/ The list of possible concepts might be further reduced by pre- and post-conditions relating to the individual "artefact" and "activity" concepts identified in the project-level model, to present the argument developer with a list of candidate terms with which to instantiate the fact types reflecting the practice of the project. More complex fact types might be devised – around the basic claim structures – to reflect complex dependencies between activities.

4 Example

In this section, we present a simple example to illustrate the ways in which structured expressions using controlled vocabulary can be exploited to instantiate claims in an assurance argument. The example is based on a simplified, fictitious automotive anti-lock braking system (ABS), which is developed to ISO 26262 [11]. Correct operation of the ABS allows the wheels to maintain contact with the road surface during hard braking, preventing the wheels from locking and avoiding an uncontrolled skid. The system comprises a software controller, four wheel sensors (one for each wheel) and two hydraulic valves (one for each axel). The system has two basic operational scenarios. The software constantly monitors the speed at which the wheels rotate, measures via the wheel sensors. If it detects that one wheel is rotating at a slower speed than the others, the controller actuates the hydraulic valves to reduce hydraulic pressure to the brake, thus reducing braking force on that wheel and allowing it to turn faster. Alternatively, if the software detects that one wheel is turning significantly faster than the others, the valves are operated to increase hydraulic pressure to that wheel, thus increasing braking force to that wheel and slowing down its rotation. The software controller contains a critical function to calculate the hydraulic pressure demand value from the wheel speed sensor inputs. Failure of this function results in the incorrect braking force being applied to the wheel, which could result in a skid.

The assurance argument for the ABS software controller clearly needs to address the issue of potential faults in the hydraulic pressure demand calculation function. In this example, that issue will be addressed as part of a top-down argument concerning the mitigation of the "uncontrolled skid" hazard by the software. An argument of this type can be structured using the approach suggested in the high-level software safety argument pattern in [10], which is presented in Figure 3, using the GSN [1]:

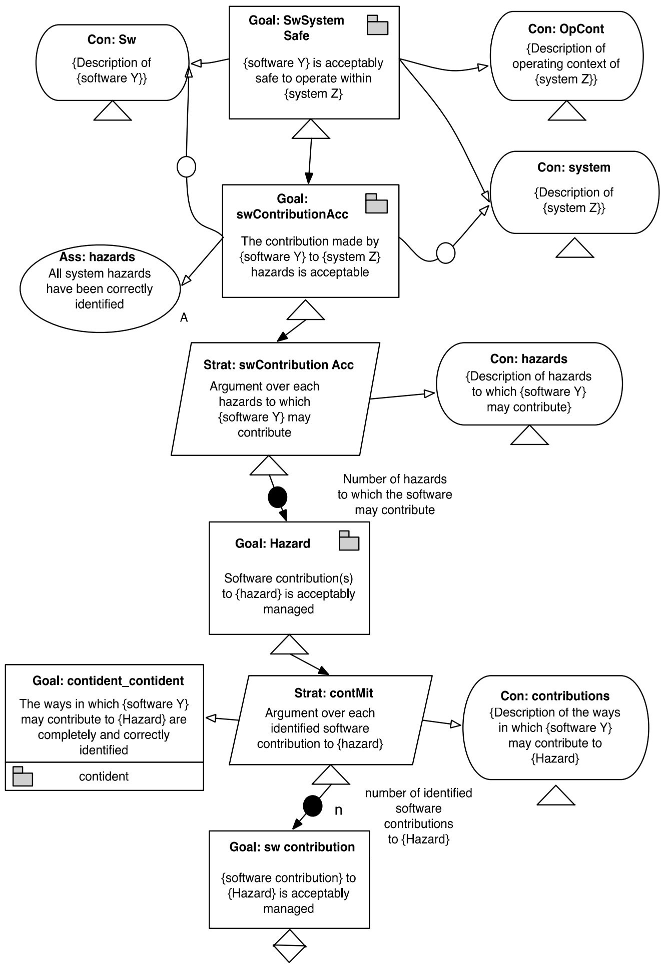

Con: Sw
{Description of {software Y}}
Goal: SwSystem Safe
{software Y} is acceptably safe to operate within {system Z}
Con: OpCont
{Description of operating context of {system Z}}
Con: system
{Description of {system Z}}
Goal: swContributionAcc
The contribution made by {software Y} to {system Z} hazards is acceptable
Ass: hazards
All system hazards have been correctly identified
A
Strat: swContribution Acc
Argument over each hazards to which {software Y} may contribute
Con: hazards
{Description of hazards to which {software Y} may contribute}
Number of hazards to which the software may contribute
Goal: Hazard
Software contribution(s) to {hazard} is acceptably managed
Goal: contident_contident
The ways in which {software Y} may contribute to {Hazard} are completely and correctly identified
contident
Strat: contMit
Argument over each identified software contribution to {hazard}
Con: contributions
{Description of the ways in which {software Y} may contribute to {Hazard}
n
number of identified software contributions to {Hazard}
Goal: sw contribution
{software contribution} to {Hazard} is acceptably managed

Figure 3 High-Level software safety argument pattern (from [10])

In the diagram, the rectangular boxes represent claims made about the software (these are called "Goals" in GSN). The top-level goal (Goal: SWSystemSafe) contains an overall claim that the software is acceptably safe to operate within the system in which it is located ({system Z}). The rounded rectangles attached by hollow arrows to this goal contain contextual statements required to further explain and validate the goal. Here, they refer to supporting documentation which provides descriptions of relevant aspects of the software design and the design and operational environment of {system Z}. The triangles underneath items indicate that textual information within curly braces requires instantiation in an argument relating to a real system. Goal:SWSystemSafe is refines into a lower-level claim (captured in Goal: swContributionAcc), which indicates that the argument will be made by considering the possible contributions that {software Y} could make to system-level hazards. The oval (Ass:hazards) represents an assumption on which this argument relies: in this case, that all of the system hazards have been identified correctly. The parallelogram (Strat:swContributionAcc) represents the strategy used to break down this general claim into more detailed ones. Here, the argument is structured by taking each of the system-level hazards to which the software may contribute in turn, and arguing that the software contribution to each has been managed. This strategy is realised in the statement of Goal:Hazard, which makes the claim that the software's contribution to a particular hazard ({Hazard}) is acceptably mitigated. An enumeration of the relevant hazards is provided as context to this argument, and is referred to in the GSN Context (Con:hazards). The solid circle on the decomposition arrow between Strat:swContributionAcc and Goal:Hazard indicates that Goal:Hazard and the subsequent argument is iterated for each of the hazards to which the software might contribute. Where a safety requirement exists which relates to the software's role in {Hazard}, this is explicitly stated, and referred to in the context Con:safetyRqt. Since software might contribute to the occurrence and effects of hazards in a number of different ways (depending on the nature of the hazard, the software and the system context), a further strategy (Strat:contMit) is applied, by which the claim concerning the safe management of these software contributions (captured in Goal:swContribution) is made and argued through for each potential contribution. This line of argument is made in the context of an enumeration of the potential contributions the software could make to the hazard (referred to in Con:contributions). Further confidence in the adequacy of the argument at this point is provided in a backing argument, which supports a claim that the list of potential software contributions to the hazard is complete and correct. This argument is made in a separate GSN module (contident), the structure of which is not outlined in full here. Goal:contident_contident provides a reference to the topmost claim in that backing argument, and serves to direct the reader's attention to the argument and evidence provided in the contident module.

Our discussion of the use of the SBVR vocabulary and claim types to develop and instantiate an argument draws on the lower part of the pattern in Figure 3, the claim in Goal:Hazard that the software's contribution to a particular Hazard is adequately managed and the subsequent argument addressing each potential way in which the software could contribute to the hazard.

The example requires two distinct SBVR vocabularies. Firstly, the ABS system is represented in a vocabulary, terms in which are drawn from the organisational vocabulary for the system as a whole. Concepts in this vocabulary serve to define concepts in the deployment context of the ABS software. The ABS software is also represented by a dedicated, project-level, vocabulary.

Figure 4 contains a restatement of the argument structure, which represents a partial instantiation of the template pattern presented in Figure 3, as an assurance argument for the ABS software. Here, Goal G1 represents an instantiation of Goal:Hazard in Fig. 3. Contexts C1 and C2 and G:backing_top are also instantiations of the parallel elements in the GSN pattern. The underlined terms here ("ABS software", "uncontrolled skid hazard", "safety requirement 123", "fault tree analysis") are instances of the more generic concept types used in Fig. 3, and are taken from the SBVR vocabulary for the ABS system (populated from project documents at the system level, such as system descriptions, requirements documents, system safety analysis).

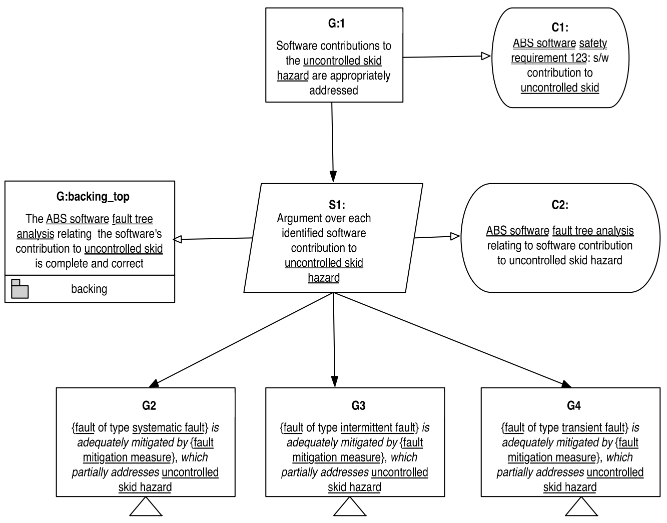
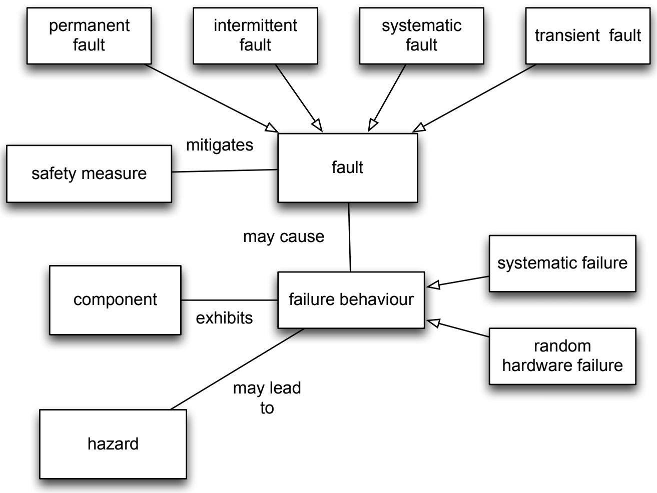

G:1
Software contributions to the uncontrolled skid hazard are appropriately addressed
C1:
ABS software safety requirement 123: s/w contribution to uncontrolled skid
G:backing_top
The ABS software fault tree analysis relating the software's contribution to uncontrolled skid is complete and correct
backing
S1:
Argument over each identified software contribution to uncontrolled skid hazard
C2:
ABS software fault tree analysis relating to software contribution to uncontrolled skid hazard
G2
{fault of type systematic fault} is adequately mitigated by {fault mitigation measure}, which partially addresses uncontrolled skid hazard
G3
{fault of type intermittent fault} is adequately mitigated by {fault mitigation measure}, which partially addresses uncontrolled skid hazard
G4
{fault of type transient fault} is adequately mitigated by {fault mitigation measure}, which partially addresses uncontrolled skid hazard

Figure 4 Restatement of lower portion of software safety argument pattern, indicating claim types

The claims captured in the statements in Goals G2, G3 and G4 represent standard-level representations of the generic claim types "Fault Accommodation Claim" and "Hazard Mitigation Claim" identified in Table 1 above. Here, they are parameterized with generic noun types drawn from the SBVR vocabulary for ISO 26262. These claims have an underlying conceptual model, which derives from ISO 26262, and relates a typology of faults to fault mitigation measures and characterises the relationship between faults and hazards[3]. This model, and the SBVR definitions for the concepts it identifies, are presented in Figure 5:



SBVR Concept Definitions

[3] Note that ISO 26262 [11] identifies an additional subtype of fault, the concept "permanent fault". This concept requires a claim of a different type from those used to handle the other fault types, and it will be more difficult to make those claims generic. In order to simplify the discussion here and focus on the use of SBVR to populate generic claims, we have excluded "permanent fault" from the illustrative example here.

fault
Definition: abnormal condition that can cause failure of an element or an item
Dictionary Basis: ISO 26262 Part 1,§1.42 (adapted)
Possibility: fault *causes* at least one failure

permanent fault
Definition: fault which occurs and then stays until removed or repaired
Dictionary Basis: ISO 26262 Part 1,§1.88
General Concept: fault

intermittent fault
Definition: a fault which occurs repeatedly and then disappears
Source: ISO 26262 Part 1,§1.42
Dictionary Basis: ISO 26262 Part 1,§1.42 note 2
General Concept: fault

systematic fault
Definition: fault which causes a failure which is manifested in a deterministic way and which can only be prevented by applying process or design measures
Source : ISO 26262 Part 1,§1.42 (adapted)
Dictionary Basis: : ISO 26262 Part 1,§1.131 (adapted)
General Concept: fault

safety measure
Definition: activity or technical solution put in place to avoid or control systematic failures and to detect or control random hardware failures or to mitigate effects of such failures which may lead to harm
Dictionary Basis: ISO 26262 Part 1 §1.110
Necessity: safety measure *includes* safety mechanism
safety measure *is specified in* functional safety requirement
Example: definition of software without the use of global variables
Synonym: means; control

failure behaviour
Definition: termination of an element's ability to perform a function as required or intended
Dictionary Basis: ISO 26262 Part 1,§1.39 (adapted)

systematic failure
Definition: failure which can be attributed deterministically to a certain cause, and which can be eliminated only by a change to the design or manufacturing process, to operational procedures, to documentation or to organisational factors
Dictionary Basis: ISO 26262 Part 1,§1.130 (adapted)
General Concept: failure
Necessity: systematic failure *is caused by* systematic fault

random hardware failure
Definition: failure that may occur unpredictably during the lifetime of a hardware element, according to some probability distribution
Dictionary Basis: ISO 26262 Part 1,§1.92
General Concept: failure
Necessity: random hardware failure *has* probability

component
Definition: element defined at an abstraction level below that of "the system", that is logically and technically separable and is comprised of more than one hardware part or of one or more software units
Source: ISO 26262 Part 3, §1
Dictionary Basis: ISO 26262 Part 1,§1.15
General Concept: element
Necessity: a component must *contain* at least one hardware part or a component must *contain* at least one software unit

hazard
Definition: potential source of harm caused by malfunctioning behaviour
of an item
Dictionary Basis: ISO 26262 Part 1,§1.56

Fact Types
fault *causes* at least one failure behaviour

failure behaviour *may lead to* hazard

systematic fault *may cause* systematic failure

safety measure *mitigates* fault

systematic failure *is caused by* systematic fault

random hardware failure *has* probability

component *exhibits* failure behaviour

hazard *has* cause

hazard *may be caused by* failure behaviour *which is exhibited by* component

hazard *has* effect

Figure 5: Conceptual model and SBVR definitions underlying the claim types defined in Figure 4

It will be clear that the first part of the claims in Goals G2, G3 and G4 have been derived straightforwardly from the conceptual model – they assert the relationship which is modelled between the "fault" and "safety measure" concepts, captured in the fact type safety measure mitigates fault. Note, however, that the claim generation is not automatic – understanding of the concepts of assurance and argumentation are required to lead to the concept of adequacy in association with fault mitigation, and thus to make the claim subjective (as the argument requires). The second part of the claim is not generated directly from a fact type or relationship, since there is no direct link in the conceptual model between the concepts of fault mitigation and the hazard. Instead, the relationship is obtained by traversing the contextual relationships between "fault", "failure behaviour" and "hazard". In order to produce an argument for a specific ABS system, the claim types captured in goals G2, G3 and G4 are instantiated by populating the parameterized noun phrases with concepts of appropriate types from the SBVR vocabulary defined for the specific ABS system – the project-level model. Figure 6 presents a partial instantiation of Goal G2:

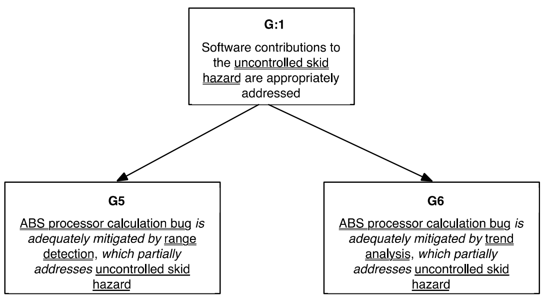

G:1
Software contributions to the uncontrolled skid hazard are appropriately addressed
G5
ABS processor calculation bug *is adequately mitigated by* range detection, *which partially addresses* uncontrolled skid hazard
G6
ABS processor calculation bug *is adequately mitigated by* trend analysis, *which partially addresses* uncontrolled skid hazard

Figure 6 Partial instantiation of claim type, using project-specific vocabulary

Here, Goal G2 from Figure 4 has been instantiated twice, populated using instances of the "systematic fault" and "fault mitigation measures' (a synonym for "safety measure") from the SBVR vocabulary for the actual ABS system (the project-level model). Note that the intention here is to show the population of the generic claim type using concrete instances from the vocabulary, rather than to present a complete argument. As it stands, the GSN fragment presented in Figure 6 suggests that the two goals G5 and G6, taken together, provide a sufficient argument that G1 holds in the context. Given the richness of the argument structure provided in Figure 4, this is clearly untrue: further instantiation of Goals G2, G3 and G4 are required to ensure adequate coverage of Goal G1. For simplicity, these additional goals (which can be instantiated from the SBVR vocabulary for the ABS system as G2 has been here) are not shown.

5 Related Work

There is only a limited amount of research which directly addresses the integration of controlled language approaches in the field of assurance argumentation. A methodology for argument development is presented in [12], which exploits the structural patterns presented in [10]. Generic patterns to help form software assurance arguments are also provided in, for example, [13], [14], [15] and [16]. Such patterns focus on the structure of the arguments and the issues they should address, rather than their phraseology or rhetoric and since they are by definition generic, it can be difficult to achieve consistency and completeness in the resulting argument instantiations. In none of these cases is explicit attention paid to the possible application of controlled natural language to enforce the patterns and assist the argument developer in making the reasoning clearer. The standard industry guidance on the development of GSN arguments [1] contains some general advice about sentence structure and a discussion of common language-based errors. These errors are identified at the level of the whole claim, rather than individual terms or phrases.

The OMG's *Structured Assurance Case Metamodel* [17] provides a metamodel of argumentation, including language aspects, and a discussion of the use of SBVR to realise assurance arguments. The technique described is, however, overly simplistic and is not fully realised in [17]: the present paper should be seen as part of an ongoing debate as to the utility of SBVR in the assurance argumentation field.

The authors of [18] define a restricted language to describe rely-guarantee conditions between software applications and computer hardware. Although this language can be used in the automated generation of a limited set of arguments concerning compositional behaviour of software elements, including failure behaviour, it is very limited in scope, and does not capture additional required information such as data concerning evidence supporting rely-guarantee claims or the degree of confidence which can be placed in them.

The OPENCOSS project [19] aims to develop technologies to support the cost-effective reuse of assurance information within and between safety-critical domains. Assurance arguments are used as the basis for communication of this information, and to support certification. This approach relies on the ability to communicate and compare relevant concepts across and within organisations and domains. However, there is no consistent conceptualisation and terminology to describe and manage assurance, let alone a "common certification approach" recognised by system integrators, the supply chain and assessors. OPENCOSS seeks to provide a basis for communication by developing a pragmatic approach, which identifies commonality and differences between the ways in which safety, assurance and certification are conceived, and provides means to compare them. The project has developed models of assurance assets, information, processes and concepts in safety standards, organisational practices and individual projects, using a generic metamodel of relevant concepts for safety assurance [6]. These models are supported by domain- and company-specific vocabularies which provide clear, controlled definitions of concepts which need to be addressed in safety arguments. A mapping technique is used to define relationships between concepts in both the models and the vocabulary at varying degrees of exactness, and tool support is provided to support engineers in making explicit the significant differences which need to be discussed in a justification of reuse.

Structured approaches to language are widely used in the requirements engineering domain. For example, the Attempto Controlled English (ACE) defines a structured natural language to support engineers in writing precise specifications which can be translated into semi-formal representations suitable for machine-checking [20]. Similarly, Denger et al [21] have identified natural language patterns to specify functional requirements for embedded systems. The CIRCE project [22] adopted model-based techniques to support the validation of natural language requirements, based on a lightweight formal model. In the safety-critical domain, the CLEAR methodology developed by the Dependability Research group at the University of Virginia uses insights from linguistics and cognitive psychology concerning the nature of linguistic error and presents a pattern-based technique to minimise miscommunication in requirements [23]. None of these methods explicitly address the issues relating to structured argumentation for assurance – for example, inherent subjectivity in claims -, although the relationship between requirements and argument claims appears to provide an interesting avenue for future research.

6 Conclusion

This paper has demonstrated the potential use of SBVR concept definitions and fact types to add rigour to the language used to convey assurance arguments for safety-critical CPS. We have described the use of a layered vocabulary and "mapping" to capture traceability relationships between concepts defined in safety standards, in organisation-specific practices and conventions and in individual projects, and have indicated how the mapping notion can be used to provide informed guidance on the transferability of concepts and the reusability of assurance assets between projects and across domains. Furthermore, we have provided an initial taxonomy of structured claim types, partially derivable from SBVR fact types, and have demonstrated how they can be used to constrain the language and focus of assurance arguments. Work to develop this method and to provide tooling is currently at an early stage. Theoretical work remains to be done to expand the taxonomy of claim types and refine their phrasing. There is also a need to explore the relationship between declarative fact types, requirements and argument claims more fully, in particular to find ways to address the subjective elements of claims in a formal or semi-formal lexicon for argumentation.

Acknowledgement

The work presented here was carried out as part of the OPENCOSS Project, No: 289011, funded by the European Commission under the FP7-ICR Framework. For further details, see the Project website: http://www.opencoss-project.eu.

References

[1] *Goal Structuring Notation Community Standard*, Issue 1 (November 2011), Available for download from http://www.goalstructuringnotation.info.

[2] http://www.adelard.com/asce/choosing-asce/cae.html.

[3] E. Lapore (2009), *Meaning and Argument: an introduction to logic through language*, Second Edition (First Edition 2000), John Wiley and Sons.

[4] K. Attwood and P. Conmy (2013), *Nuanced term-matching to assist in compositional safety assurance,* First International Workshop on Assurance Cases for Software-Intensive Systems (ASSURE 2013).

[5] Object Modelling Group (2008), *Semantics of Business Vocabulary and Business Rules*, Version 1. Available for download at http://www.omg.org/spec/SBVR/1.0/.

[6] OPENCOSS Consortium (2013), *Common Certification Language: Conceptual Model* (Version 1), Project deliverable D4.4. Available for download at http://www.opencoss-project.eu.

[7] IEC (2009), *IEC 61508: International Standard – Functional safety of electrical/ electronic/ programmable electronic safety-related systems.*

[8] RTCA (1992), *RTCA/DO-178B: Software considerations in airborne systems and equipment certification.*

[9] R. Hawkins, T. Kelly, J. Knight and P. Graydon (2011), *A new approach for creating clear safety arguments,* in C. Dale and T. Anderson (eds) *Advances in Systems Safety: Safety-Critical Systems Symposium (SSS 11)*, Springer-Verlag, pp 3-24.

[10] R. Hawkins and T. Kelly (2013), *A Software Safety Argument Pattern Catalogue*, University of York Department of Computer Science Report YCS-2013-482. Available for download at ftp://ftp.cs.york.ac.uk/reports/2013/YCS/482/YCS-2013-482.pdf.

[11] ISO/FDIS (2011), *ISO/FDIS 26262 International Standard – Road Vehicles, Functional Safety.*

[12] R. Hawkins and T. Kelly (2010), *A systematic approach to developing software safety cases,* Journal of System Safety, vol 46 no. 4, pp 25-33.

[13] T. P. Kelly (1998), *Arguing safety – a systematic approach to managing safety cases,* D.Phil Thesis, University of York.

[14] R. A. Weaver (2003), *The safety of software – constructing and assuring arguments,* PhD Thesis, University of York.

[15] W. Wu (2007), *Architectural reasoning for safety-critical software applications*, PhD Thesis, University of York.

[16] Industrial Avionics Working Group (2012), *Modular Software Safety Case Process Description*. Available for download at https://www.amsderisc.com/p-content/uploads/2013/01/MSSC_201_Issue_01_PD_2012_11_17.pdf.

[17] Object Modelling Group (2013), *Structured Assurance Case metamodel (SACM)*, Version 1. Available for download at http://www.ormg.org/spec/SACM/.

[18] B. Zimmer, S. Bürklen, M. Knoop, J. Höfflinger and M. Trapp (2001), *Vertical safety interfaces – improving the efficiency of modular certification*, in U. Voges (ed), *Computer Safety, Reliability and Security SAFECOMP 2001,* LNCS 2187, Springer-Verlag, pp 29-42.

[19] http://www.opencoss-project.eu.

[20] N. E. Fuchs, U. Schwertel and R. Schwitter (1999), *Attempto Controlled English – not just another logic specification language* in P. Flener (ed) (1999), *8th International Workshop on Logic-Based Program Synthesis and Transformation 1999*, LNCS 1559, Springer-Verlag, pp 1-20.

[21] C. Denger, D. Berry and E. Kamsties (2003), *Higher-quality requirements specifications through natural language patterns*, IEEE Conference on Software: Science, Technology and Engineering, pp 80-90.

[22] V. Abriola and V. Gervasi (2006), *On the systematic analysis of natural language requirements with CIRCE*, Automated Software Engineering, vol 13 no 1, pp 107-167.

[23] K. S. Hanks, J. C. Knight, E. A. Strunk and S. R. Travis (2003), *Tools supporting the clear communication of critical application domain knowledge in high-consequence systems development,* in S. Anderson, M. Felici, B. Littlewood (eds), *Computer Safety, Reliability and Security SAFECOMP 2003*, LNCS 2788, Springer-Verlag, pp 317-330.

Dependable Real-Time System and Mixed Criticality: Seeking Safety, Flexibility and Efficiency with Kron-OS

Vincent DAVID, Adrien BARBOT, Damien CHABROL

Krono-Safe, 86 rue de Paris, 91400 Orsay, France; Tel: +33 1 77 93 21 59; email: contact@krono-safe.com

Abstract

Embedded real-time systems integrate more and more real-time application functions on the same execution unit with heterogeneous real-time requirements but also dissimilar safety requirements. It is not realistic to apply the highest safety level to all functions, which leads to the problem of mixed-criticality. Hypervisors seem to have become a popular solution, but they consider real-time features as a secondary issue. Their main drawback is the difficulty (or impossibility) to manage different time-scales and jitters as a real-time operating system is supposed to. To cope with this problem, we propose an approach that we briefly introduce in this article. Kron-OS is a software suite to design, implement and execute real-time solutions mixing strong real-time requirements along with low-criticality features. It also provides a set of automatic code generation tools and a safety-oriented real-time kernel that includes temporal and spatial partitioning methodology and mechanisms.

Keywords: dependability, mixed-criticality, safety, scheduling, real-time, design methodology.

1 Introduction

Industrial companies want to mix functions with various requirements concerning real-time features but also different levels of criticality [4]. A better integration of tightly coupled functions can offer a better flexibility in development and a reduction in hardware costs. It should also ease the implementation of efficient communication and synchronization, but it also increases the number of potential malfunctions on the same execution unit, with the possibility to have a global impact on the system (e.g. complete shutdown). Of course, this hazard must be avoided.

Safety levels are classified depending on the considered industrial domain (Safety Integrity Levels, Design Assurance Levels, etc.), but a classification is not enough when some hardware resources are shared: the highest safety level is always the one that has to be applied because it is not acceptable that a lower-level function would have an impact on a higher-level one. In most cases, the majority of functions in a system are not safety related and as such are often called "best effort"; only a minority of functions is classified as "safety functions". Nevertheless, the term "best effort" is sometimes misunderstood: the real-time requirements of the different functions are not so different in nature (after all, real-time is real-time) whatever their level of safety is, but the level of guarantee about the real-time behavior is different. The term "best effort" only means that we accept in advance that sometimes, some real-time constraints could be relaxed. But it does not mean that things may run out of control: on the contrary, the system design must be prepared to manage hazardous situations and always recover to a safe state. And this point underlines one of the typical requirement of a mixed-criticality system: if the whole system is not correctly sized, the critical functions shall always be able to use their required resources, whereas the non-critical functions may suffer of a lack of resources. In both cases, the real issue is real-time. If there is no real-time constraint, any system with partitioning mechanisms could be an acceptable solution.

Thereby, an execution platform shall provide spatial and temporal partitioning mechanisms, but it should also provide abilities to manage real-time in a formal way (multiple timescales and jitters). As said by Edward A. Lee [5] and many others, current asynchronous kernel technologies are inappropriate to manage the system increased complexity. Because of non-determinism and uncontrolled temporal behaviors, current systems have a low level of testability that leads to many long tests campaigns, which completeness is complex (almost impossible in practice) to achieve and to demonstrate. Moreover, spatial and temporal partitioning mechanisms are often based on a poor confinement granularity, which leads to late error detections. Non-interference is complex to ensure without degrading performance or usage flexibility. Hypervisors enforce the spatial and temporal segregation between partitions [6], with many variants and trade-offs depending on the level of interference that is acceptable, but this approach introduces two or more hierarchical scheduling levels, with prohibitive costs and poor real-time performance for the vast majority of real-time systems.

This article focuses on the temporal and spatial partitioning principles of the Krono-Safe technology. This technology addresses the issue of determinism in real-time and mix-criticality systems.

2 Historical approaches

In the aeronautical domain, following the application of DO-178A standard [7] and its Design Assurance Levels in the eighties, the system suppliers, the aircraft manufacturers and the certification authorities had to agree upon the safety levels for each function of the system. Since the quantity of activities (tests, demonstrations, with or without independence) to perform in order to reach a DAL level is very different when the function is at level A (highest) or at level D (lowest for embedded software), both system suppliers and the aircraft manufacturers had in mind to lower the level as much as possible thanks to the system architecture (redundancy, votes, etc.). However, the system architecture could not lower the level of all functions on the aircraft, so some remain at a high level (e.g. control loops at level A or B), when other could be at lower level (e.g. maintenance functions at level C). Then the system suppliers had a choice between developing a single hardware with all software components at the highest DAL or developing several hardware with a dedicated software on each one.

For most system suppliers and in particular for small aircrafts, developing everything at the highest DAL was acceptable as long as the system were not very complex. However, because of the competition between aircraft manufacturers, it became necessary to include more and more "comfort" functions in the systems (auto-brake, fuel consumption improvements, etc.) which come in addition to the core functions.

In the nineties, the aeronautical industry decided to address this problem of mixed-criticality by developing the Integrated Modular Avionics concept. The idea was to rely on standardized platforms called "modules" (hardware and operating system) developed at the highest level of safety, and these modules would propose spatial and time partitioning mechanisms that guarantee the non-interference of application functions. This approach led also to the standardization of communication buses between the modules and the API of the operating systems. Several ARINC standards were defined at that time, including the ARINC 653 for the operating system [8].

The ARINC 653 addresses several problems behind the mixed-criticality issue: the incremental certification (being able to certify part of the application functions), the definition of roles for an industrial breakdown structure that corresponds to the aeronautical industry (manufacturer, system supplier and function suppliers) and the way to actually mix the functions of different DAL on the same hardware.

In the ARINC 653 approach, a "module integrator" is responsible for the allocation of resources on each module of the system. This includes both spatial (ROM and RAM) and timing (periodicity of treatments, budget allocations for each partition) aspects. Usually, the module integrator defines the allocation based on the needs expressed by the system suppliers and the resources available on the hardware. In most cases, this allocation is manual, meaning that the system suppliers provide their needs as they can (e.g. "5ms of CPU time every 40ms, 600KB of RAM"), and the module integrator needs to find a way to answer the needs of all the partitions on the module ("two slots of 2.5ms in 40ms, RAM between 0xFF000000 and 0xFF96000"). When the requirements cannot be met, a trade-off is required to reach a compromise on the module so that every partition has enough resources. We insist on the fact that, up to this day, this often remains a manual activity.

This is very time consuming (it may take up to 6 months to reach this compromise on a single module in a big aircraft program), and it has to be performed for every software version of every function on the module, because the needs may change in time (addition or removal of a function). The safety of the whole system and the ability of a module to perform a safety function (and all other functions for that matter) rely entirely on this allocation, and since they are manual and based on non-formal requirements, the process is prone to errors.

Around the same time, the French Atomic Energy Commission (Commissariat à l'énergie atomique - CEA), being aware of the work of the ARINC consortium and facing the same challenges in the nuclear industry, decided to propose another approach, which is the ancestor of the Krono-Safe technology. The idea was to create a semi-formal language to express the timing constraints, and that the allocation would be automatically computed based on these constraints. The same principles would be applied on the spatial requirements based on the actual needs of the software (by analyzing the compiled binary) and allocating automatically the resources in ROM and RAM, and configuring automatically the hardware mechanisms needed to ensure the safety of the system (e.g. MPU configuration). These principles were industrialized in OASIS for the nuclear industry [10], and developed as a proof of concept in the automotive industry under the name "PharOS". Kron-OS, developed by Krono-Safe, is based on the same principles as the OASIS technology.

3 Non-interference

Once the partitioning system is implemented, the real problem begins: how to guarantee the non-interference of functions, for space (no data or code access between partitions) and time.

For space partitioning, there are only two solutions: to use a hardware device for protection (Memory Management Units or Memory Protection Units), or to catch every single access to the memory by software. We exclude the formal proof of software because it is only limited to simple and small source code at the time of writing of this article.

The software solution is applied in hypervisors, often relying also on hardware for address translation to speed up the process, and by construction it leads to slow access times (more instructions to execute for the same access). This requires a lot of computing power, and can seriously be considered only for high-end CPUs.

The full-hardware solution is used for safety-related systems because of its speed and its independence from the compiler which eases the safety demonstrations; however the hardware configuration can be complex and not optimal: e.g. with most MPU's the memory is divided into "pages", and sometimes these pages must be aligned on sizes that are powers of 2, leading to unused holes in the memory.

Moreover, to protect certain particular areas, such as the execution stacks, the neighbors of a given page must comply with additional constraints (e.g. data of a partition cannot be located next to the stack of this partition to enable the detection of stack overflow).

Obviously, a human being with a lot of spare time can find a solution (maybe not optimal, but it will work); but this is not realistic for industrial applications: every time the memory requirements changes, the whole allocation has to be done from scratch.

The topic of non-interference for time is more complex because of the languages used for real-time programming, mainly: assembly, C and Ada. Behind non-interference for timing aspects, Krono-Safe addresses also the following issues:

- The determination of budgets of time for each function on the system, in particular with an incremental process where each partition is estimated alone on the hardware and then integrated with all other partitions with the same properties;
- The allocation of slots of CPU time that take into account error behaviors in any function without impact on any other software component on the same hardware;
- The ability to determine which task has exceeded its allocated budget, and the ability to give it some additional time if there is some left on the module without any interference with other partitions.

The determination of the time budgets required by a task in a real-time system has always been a problem, and neither C nor Ada have solved the issue in the programming language. In fact, the problem cannot be solved at that level because the transformation from a high-level language to binary instructions is very complex, and because the hardware itself may have non-deterministic behaviors in its treatments (one of the reasons why offline CPU execution estimations tools work so badly, with often 3 to 5 times overestimations).

For example, depending on the preemption point in a partition, the next partition in the scheduling plan will start with its cache in any state. The duration of partition which is preempted is therefore longer than it would be without a preemption point (see Figure 1).

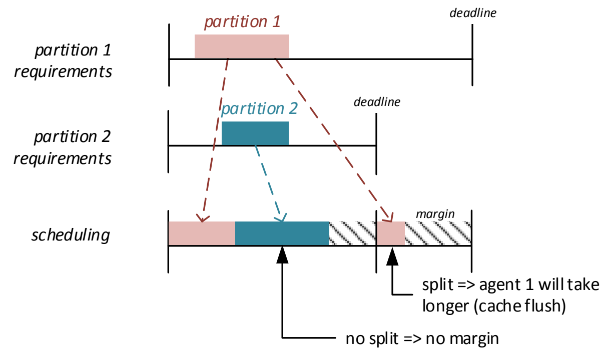


Figure 1. Impact of a split on the execution time

The easiest solution is to flush all cache lines at each preemption, but this has a major impact on performance as modern CPUs rely heavily on cache to maintain the speed of execution when the memory buses are two or more times slower than the cores.

Provided enough project time and resources are available, the manual activities may eventually work. They have been applied successfully for some aircrafts in the past 20 years. Krono-Safe thinks that the development costs may be reduced and the hardware optimized with adequate automatic tools, and we will show this in a later paragraph. The next problem however cannot be addressed with offline scheduling policies: non-interference of error treatments.

During the execution of some source code, every single instruction could lead to an error (division by zero, memory access violation, corruption of code or data, etc.). This means that for every single instruction in the software, additional time has to be considered to deal with the error and to take an appropriate action.

In the ARINC 653 approach, the standards states that "Temporal partitioning is influenced by the O/S overhead. Inter-module communications acknowledgements and time-outs may interrupt one partition even though the events relate to a different partition. As a result, the time duration allocated for use by an application may be impacted" (from [265]). So it is up to the module integrator to put some spare time slots between partitions so that if a partition has an error at the end of its allocated slice, it does not delay the start of the next partition.

When the module integrator does not plan for these slots (or does not have spare time for the module), then a jitter has to be considered, sometimes called "slice-out time", which corresponds to the duration of the longest non-interruptible service in the system that will be executed on error. This jitter applies to the next partition in the scheduling plan, which means that non-interference is absolutely not enforced in this kind of technology.

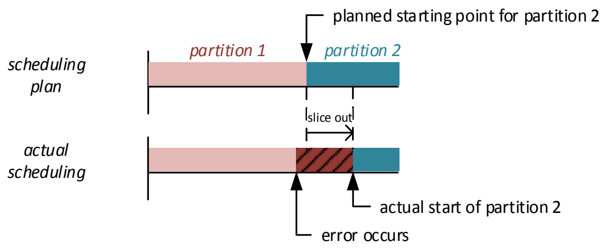


Figure 2. Slice out impact on scheduling

In Figure 2, if an error occurs in partition 1 near the end of its allocated time window, the operating system enters a non-interruptible service to deal with the error (also called "critical section"), therefore delaying the start of partition 2. This delay has to be taken into account in the allocated time window of partition 2 so that it does not propagate further than this windows. The usual way to deal with this issue is to add a margin to the required time of partition 2, which means that partition 1 (or in fact, the scheduling policy) has an impact on partition 2.

When spare time slots are allocated so that the error management treatment of one partition does not impact the next one in the scheduling plan, the main issue is that these slots are lost in the nominal case when everything is normal on the computer. This leads to oversized hardware, used only at 50% or less as long as nothing goes wrong. Again, Krono-Safe thinks that this approach is not acceptable in the long-term (additional constraints to the scheduling plan which is already complex to build) and that the spare resources should be used as long as the non-interference principles are enforced (especially in industrial domains where cost reductions lead to small CPU's).

This leads to the last point addressed by Kron-OS that is not covered by any technology on the market today: the ability to give some additional time to a partition while preserving non-interference with other partitions execution. In a static approach like ARINC 653, where a module integrator manually allocates resources to the partitions and all demonstrations are made from this scheduling plan, it is not possible to decide anything during the execution of the plan. This means that if a slot is reserved for idle time (margin), it cannot be used to give some extra time to one of the partitions (e.g. if the allocated budget was underestimated) as it would break all demonstrations of non-interference.

On all these points, for both space and time partitioning, the Krono-Safe technology proposes innovative concepts that are automated, keeping in mind that every step needs to be qualified in the scope of industrial standards.

The main advantages of the Kron-OS approach are:

- The formal specification of both spatial and timing requirements, which are not subject to interpretation by a human;
- The automatic generation of scheduling and memory tables that answer the requirements (correct by construction) so that there is no additional development cost when the specifications are modified during the development;
- A simple real-time scheduler that can take fast decisions and determine the agent at fault so that other agents are not impacted (for both time and space) without requiring an oversized hardware module.

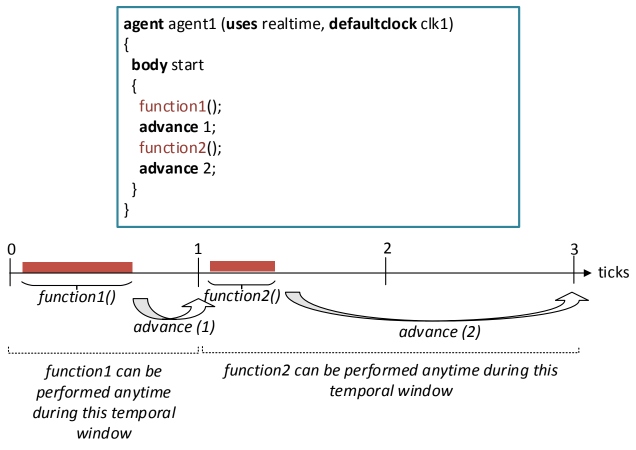
4 The Kron-OS Safety Approach

In all industrial domains, designing a "safe" system means reaching an acceptable level of confidence in the functions performed by the system. To this end, a safe system must be proved by construction and a software safety demonstration must rely on:

- A design approach based on a multitasking programming model which enables demonstration (e.g. temporal behavior, communication, synchronization) and
- Controls performed at runtime in order to guarantee an execution in conformance with design and safety requirements (e.g. spatial and temporal partitioning).

In Kron-OS, the first item is covered by a formal design methodology for real-time systems that is supported by a semi-formal programming language created by CEA and improved by Krono-Safe, the Psy language (Parallel SYnchronous language), and a complete tool chain that includes a compiler and an automatic code generator. For historical reasons, the first implementation of the Psy language had been based on the C language, called PsyC. The PsyC relies only on the control statements of the underlying language (if, for, while, etc.), so it can be easily adapted to Ada or any other language, and this has been demonstrated successfully in a mock-up as a proof-of-concept on Adacore compiler GNAT.

In PsyC, the main parallel executable entity is called "agent". It has its own execution context: time (deadlines for treatments, allocated budget times) and space (ROM and RAM allocation). It is possible to specify the agents' real-time behaviors with timing and dataflow descriptions, even with a mix of periodic and aperiodic activities, as briefly illustrated in the Figure 3:



```
agent agent1 (uses realtime, defaultclock clk1)
{
 body start
 {
  function1();
  advance 1;
  function2();
  advance 2;
 }
}
```

0
1
2
3
ticks
function1()
function2()
advance (1)
advance (2)
function1 can be performed anytime during this temporal window
function2 can be performed anytime during this temporal window

Figure 3. Example of timing design with Kron-OS

The main benefits of this approach are:

- Safety: dataflow consistency is guaranteed thanks to a specific communication mechanism; the application behavior is predictable and reproducible, and freedom from interference is guaranteed even in case of failure;
- Flexibility: the timing design issue is disconnected from the scheduling and optimization issues, so the side effects of a timing change at high level are bounded, the system can use periodic/aperiodic treatments, and it can use both time-triggered and external-triggered sources [11];
- Efficiency: preemptions points and interruptions are reduced to a minimum thanks to a specific scheduling strategy (frames scheduled by a tick from an internal or external source, which is a significant improvement of [12]), end-to-end constraints and jitters are guaranteed by construction, buffers and stacks sizes are statically defined, and the scheduling is locally optimal.

To specify the temporal behavior of the agents, time is manipulated in PsyC as “clocks”, which are sets of formal instants in time, called “ticks”. Agents are then divided into elementary actions cadenced by these ticks: a treatment in PsyC has an earliest start date (the treatment can start no sooner than a certain tick) and a deadline (the treatment cannot continue after another tick). Within the time slots defined by this points in time, Kron-OS is free to organize the agents as long as the constraints are enforced.

This level of abstraction enables the user to specify the needs in a formal way, independently from the final hardware or the environment. Then, the Kron-OS tool chain can compute a scheduling plan in line with all the constraints and needs expressed in PsyC.

When the user wants to execute the code on a real target, s/he has to specify the needs in terms of time budgets. These budgets can be estimated in a number of ways: Worst-Case Execution Times (WCET), engineers’ estimates, lessons learnt from past project, etc. They depend on the CPU power, so the user expresses separately his needs in terms of cadence and in terms of execution. This means also that changing the hardware, even if it is late in the development, consists only in providing new budgets for the same agents cadence and producing a new scheduling plan with these values.

The user’s needs are not always periodical, e.g. the partition may be in a functional “active” mode with a period of 5 ms and in a “passive” mode with a period of 1 second. In PsyC, the timing constraints can be conditional, based on the C language for expressing these conditions (if, for, while). Krono-Safe has patented a computation process to produce Repetitive Sequences of “Frames” (time intervals for each agent), or “RSF”, from any cadence of agents. The main advantage is that however complex the cadence is, the resulting sequence is always finite and bounded (see Figure 4).

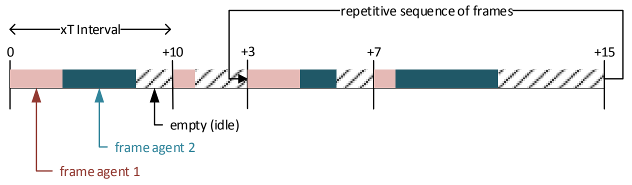


Figure 4. Repetitive Sequence of Frames (RSF)

The embedded scheduler is then responsible for following the RSF, with many benefits: no dynamic scheduling, better control of margins than EDF (Earliest Deadline First), less preemptions than EDF, optimal CPU load smoothing with multiple tasks and multiple rhythms and time-scales.

It should be noted that RSF can use time-triggered (internal periodical time-based ticks) or external-triggered sources (acquired physical signal, arrival of a frame on a network, etc.) [7]. It should be noted also that since time is manipulated as an abstraction at PsyC level, it is possible to simulate the behavior of all agents on a computer for early design validation. This simulation is fully representative of the temporal behavior on the final hardware (as long as the hardware is powerful enough to execute the instructions).

5 Multi-RSF Mechanisms for Mixed-Critical Functions

As we briefly introduced above, Kron-OS tool chain is able to generate automatically the configuration tables that describe the temporal and spatial partitioning used by the Kron-OS kernel: a strict spatial access policy has been defined and implemented in conformance with MILS architecture in order to isolate precisely each component and to design a deterministic multitask system. This protection is implemented with a memory protection unit, and thanks to a specific binary segmentation, each task accesses only to memory areas with its sufficient and necessary rights.

The method used to compute the RSF leads to the identification and the optimal division of residual margins. For example, this feature can be used to enable error recovery (and more generally health monitoring) within a deterministic framework in real-time and without interference with other tasks. This improves the robustness of multitasking real-time systems and reduces drastically residual errors due to execution budgets overrun.

This method is also useful to segregate critical and non-critical tasks because a RSF may be (and actually is) built incrementally: two (or more) RSF can be combined in order to produce a “single” RSF that merges all frames of all tasks, in compliance with real-time requirements. This idea is simple but powerful: the RSF structure guarantees the non-interference between agents, thanks to the temporal and spatial segregation of each frame, whatever a frame is: a slice of a critical task or a slice of non-critical task. Scheduling is achieved at the frame level to satisfy highly constrained real-time systems (and not with two hierarchical scheduling levels such as partitions and processes inside partitions).

Understanding that point, it becomes easy to design a whole RSF as the compound of two RSF optimally interweaved, one for the subset of critical tasks and the other one for the subset of non-critical tasks. Then, the only difference in the tasks' management will be the strategy for error recovery. For the non-critical part, a.k.a. "best-effort tasks", in case of a possible overload, a treatment can be postponed to the next available frame in the set of non-critical tasks: the time will just shift but the order in the sequence will be the same. Budgets are still ensured for the critical part because of the guarantee of scheduling correctness, as a result of configuration table and the guarantee of freedom from interference thanks to the independence of timing. For the non-critical part, the time-shift is monitored by the Kron-OS kernel and is limited to a known bound, which is dependent on the application.

For multi-core applications, one or more RSF are produced by core, and the Kron-OS kernel is responsible for the communication between the cores. It should be noted that no core synchronization is required, and only time causality has to be enforced: one agent can only access data produced before its earliest start date, and cannot produce data for other agents before its deadline. In this approach, there is no need to dedicate one core to safety functions and another one to less critical tasks.

From a performance standpoint, the main advantage of the RSF is that the preemption points are known in advance, even when the agent's cadence is not periodical. This means that it is possible to add a "preemption margin" to the time budget expressed by the user each time the treatment can actually be preempted.

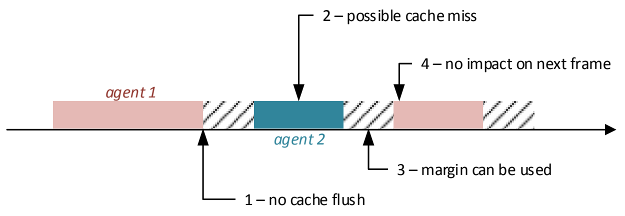


Figure 5. No useless cache flush

In the Figure 5, there is no need to flush the cache at the end of the agent 1 frame (1) because even if agent 2 takes longer than budgeted (cache miss in 2), there are some margins for that in the RSF (3) that can be used to provide some additional time to the frame of agent 2. Then the next frame starts on time (4), as planned.

This is different from the ARINC 653 approach where the scheduling is computed manually based on the user's budgets expressed in a non-formal way, in which case it is up to the user to add some margins (without any knowledge about the final scheduling plan), leading to very pessimistic budgets requirements.

Krono-Safe approach consists in a practical approach to a very common issue in real-time programming: expressing timing constraints independently from the hardware, and then configuring this hardware in accordance with all these constraints. With the increasing complexity of embedded systems, these activities can no longer be performed by humans alone: they must be assisted with tools that enables them to focus on the problem, not on solving a very complex constrained problem.

6 Implementation

The Krono-Safe tool suite is currently under development, but it already successfully demonstrated the principles and the gains expected with the technology. The suite itself, called KRONO-SUITE (temporary name), is composed of:

- An IDE for PsyC source code writing and the PsyC compiler; the PsyC compiler translates the PsyC code into C, which in turn is compiled and linked with a target or host compiler/linker;
- A simulator on host with a trace system, which enables the user to see on chronograms the timing behavior of his agents;
- An embedded kernel which implements an RSF scheduler and all protection mechanisms to reach the functional safety.

The tools proposed by Krono-Safe can be adapted to any hardware and any compiler on the market as long as it proposes a set of minimal features (memory protection hardware, real-time timers, ability to control the linking process in particular).

The kernel has successfully be ported on several targets, including an ARMv7 (8MHz, 96KB ROM and 16KB RAM) and an Infineon TriCore CPU with 3 cores (200MHz, 4MB ROM and 120KB RAM).

On the ARM CPU, a scheduler with two RSF's has been implemented: one for the critical agents (meaning that the timing aspects have to be strictly enforced) and another one for the non-critical agents ("best-effort" agents for display). The critical RSF is cadenced by an external trigger, an ASIC dedicated to providing a time source for the CPU; the non-critical RSF is based on time and is executed only when the critical one is idle.

The kernel can take several actions when the non-critical RSF does not enforce its deadlines: as long as the delay is acceptable (configurable by the user), the kernel continues to execute the agent; then when the delay is above the thresholds, a recovery action is applied (e.g. stop the RSF).

We see on the following screenshot that the non-critical RSF is slowed down when the CPU load is too high:

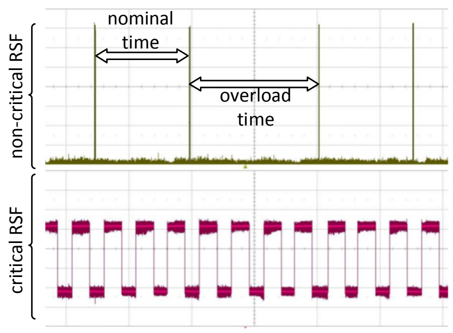


Figure 6. Critical/Non-critical RSF mix

For information, on this ARM (again, at 8MHz), the critical RSF is executed with a period of 5ms, interrupted every 130µs by an interrupt coming from the external ASIC, and the non-critical RSF is executed with a period of 10ms. This port of Krono-Safe includes the protection mechanisms for both time and space (using the MPU).

On the TriCore CPU with 3 execution cores, since the hardware implements a data cache, it has been possible to assess the impact of RSF re-organization according to criteria defined by the user. For example, by putting in sequence the frames allocated to the same agent, it is possible to gain up to 3% on the cache misses. Krono-Safe is investigating other leads for the optimizations, such as:

- To remove useless pre-emptions: the mechanism, called "anticipation", consists in executing in sequence two frames without pre-emption as long as there is no deadline on this point;
- To increase of idle time to improve the power saving features of the CPU;
- To adjust memory protection areas (groups of agents protected as a whole) so that the memory protection configuration is not changed at each agent.

The implementations of the kernel and the tool chain are still being optimized for the targets; however the results are promising and the ability to manipulate the RSF according to user's criteria (more idle time, more cache optimization, etc.) is a powerful tool to optimize the use of the hardware.

7 Conclusion

The method briefly introduced in this article is dedicated to the design of real-time system with mix-criticality functions with the benefits of safety, flexibility and efficiency. All these mechanisms are integrated in Kron-OS, a complete software suite including a kernel running on single-core and multi-core architectures, and associated support tools providing the breakthrough to organize the runtime with RSF, compared to traditional approaches (asynchronous kernel, and/or hypervisor). The provided protection mechanisms ensure early errors detection and confinement at task level. Thus, it is possible to define failure management strategies in order to improve availability and real-time performance without degrading safety. These works have been validated on several industrial use-cases, Kron-OS being currently industrialized by Krono-Safe.

The possibility to have a better integration of functions on the same core is an opportunity to decrease the number of hardware modules in a system and as a consequence also the overall system power consumption. Thanks to a combination of a kernel and an automatic code generation tool, Krono-Safe offers an innovative and complete product suite dedicated to the development of safe applications in time-to-market constraints, and is offering new perspectives for cyber-physical system/real-time systems.

References

[4] Report from the Workshop on Mixed Criticality Systems, cordis.europa.eu/fp7/ict/embedded-systems-engineering/documents/sra-mixed-criticality-systems.pdf

[5] Edward A. Lee (2009), *Computing Needs Time*, Technical Report No. UCB/EECS-2009-30, www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-30.html

[6] Alan Burns and Rob Davis, *Mixed Criticality Systems - A Review*, Department of Computer Science, University of York, York, UK, www-users.cs.york.ac.uk/~burns/review.pdf

[7] RTCA (1985), *DO-178A Software Considerations in Airborne Systems and Equipment Certification*.

[8] ARINC (1996), *ARINC Specification 653, Avionics Application Software Standard Interface*.

[9] *ARINC Specification 653, Avionics Application Software Standard Interface Part 1 – Required Services*, A653P1-3, November 15, 2010, ARINC.

[10] V. David, C. Aussaguès, S. Louise, P. Hilsenkopf, B. Ortolo, and C. Hessler (2004), *The OASIS Based Qualified Display System*, in 4th American Nuclear Society Int. Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interface Technologies (NPIC&HMIT), Columbus, Ohio, USA.

[11] D. Chabrol, V. David, P. Oudin, G. Zeppa and M. Jan (2014), *Freedom from interference among time-triggered and angle-triggered tasks: a powertrain case study*, ERTS^2 2014, Toulouse, France.

[12] C. Aussaguès and V. David (1998), *Guaranteeing Timeliness in Safety-Critical Real-Time Systems*, IFAC 15th Workshop on Distributed Computer Control Systems (DCCS'98).

Behavioral Contracts for Energy Consumption

S. Nakajima
National Institute of Informatics, Tokyo 101-8430; Tel: +81 3 4212 2507; email: nkjm@nii.ac.jp
M. Toyoshima
DENSO CORPORATION, Aichi 448-8661; Tel: +81 566 61 4770; email: MASUMI_TOYOSHIMA@denso.co.jp

Abstract

The energy consumption is one of the major non-functional concerns for systems with limited battery capacity. An application program, although functionally correct, may suffer from unexpected energy consumption. Such energy bugs (ebugs) are detected at runtime. Some ebugs, however, are desirable to remove at early stages of the software system development because they are design faults. This paper studies the energy consumption problem in the Android applications of smartphones or tablets, and presents a formal model of the energy consumption behavior that can be a basis of model-based analysis methods.

Keywords: Energy Bugs, Smartphones, DVFS, Hybrid Automata.

1 Introduction

The capacity of battery in smartphones or tablets is limited and the energy consumption is one of the major non-functional concerns to be carefully examined at early stages of software system development. An application program, even if functionally correct, may suffer from unexpected energy consumption, which is called *energy bugs (ebugs)* [11]. Although the hardware components are direct consumers of the battery, the application program using these platform components should be responsible for the ebugs. Faults caused by such ebugs are, in practice, checked up running programs by energy profilers (cf. [12] [14]). The approach, however, has several drawbacks; (a) checks are conducted by running programs although some root causes are originated from design flaws, and (b) the coverage is limited by the supplied test execution data or test environment setup.

A model-based method for examining the energy consumption phenomena is desirable to counter the disadvantages of the profiler-based method, and an appropriate abstract *model* plays a key role. One such formal model was proposed in [8] to account for the energy consumption behavior. The model, power consumption automaton (PCA), is defined as a subclass of linear hybrid automaton [5]. It is, however, not clear how the PCA incorporates platform-dependent aspects. Since hardware components are direct consumers of the battery power, the energy consumption of application programs cannot be platform-independent. As such a platform-dependent aspect, this paper studies the effects of power-saving processors, the dynamic voltage-frequency scaling (DVFS) [7]. A

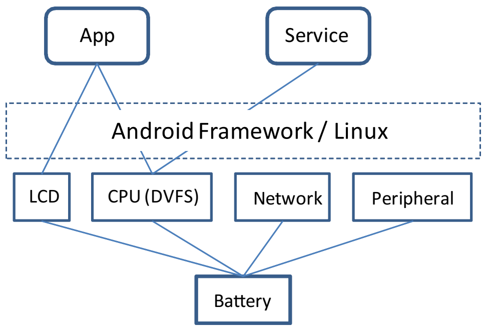


Figure 1: Android Architecture

recent paper [6] reported that the total energy consumption of smartphones could be reduced by choosing an optimum operation frequency. This observation showed that the energy issue was platform-dependent. The definition of the PCA in [8] may be changed accordingly, and quantitative arguments are important to examine such an extended PCA model.

The contributions of this paper are as follows. (a) We obtained some quantitative results of the platform-dependent effects on the energy consumption. (b) We introduce a probabilistic variant of the PCA to take into account the platform-dependent aspects. Furthermore, we show how the model-based and profiler-based methods are complementary in the problem of detecting energy bugs although it is rather our conjecture than a definitive answer.

2 Energy Consumption Issues in Android

Figure 1 illustrates an abstract view of the Android-based architecture [1]. It focuses on the components that are related to the battery power consumption. Application processes, either App or service, use devices such as networks or peripherals. These hardware components are direct consumers of the battery power[1].

The Android framework encapsulates the underlying components and provides appropriate abstractions for programmers. The multi-layered hierarchical architecture makes it difficult to understand the energy consumption behavior precisely. While hardware components are direct consumers of the battery power, the consumption is attributed to application programs. The programs invoke methods to control the usage of hardware components such as Wi-Fi or GPS. These

[1]A docking station is a *supplier* connected to the external power source. This paper considers the consumers of the battery.

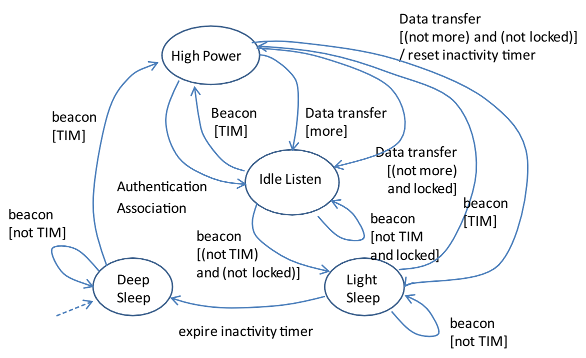


Figure 2: PCA of WiFi Client with WifiLock Control

components may result in tail-state energy consumption in which case finding root causes in a program is difficult.

Although several techniques are devised, such as the system call tracing [10] or taint analysis [14], debugging ebugs of Android application programs is hard just by using runtime profilers. An alternative method to use abstract models of energy consumption is desirable to complement the profiler-based method.

3 Power Consumption Automaton

We recall here the power consumption automaton (PCA) introduced in [8]. A simple example is shown in Figure 2, a diagrammatic form of a PCA for a WiFi client with a WiFiLock. It is a state-transition system, in which each state is called *power state* to consume energy at a particular rate, and state-transitions include timer timeout.

The power consumption automaton (PCA) is a 6-tuple. The definition follows the presentation in [5] so as to make it easy to compare with the linear hybrid automata (LHA). A PCA is, indeed, a strict subclass of LHA.

$$\langle\ Loc, Var, Lab, Edg, Act, Inv\ \rangle$$

The components are explained below.

1. Loc is a finite set of locations to represent the power states.

2. Var is a finite set of real-valued variables. A valuation v for the variables is a function to assign a real-value $v(x){\in}R$ to each variable $x{\in}Var$. V represents the set of valuations ($v{\in}V$).

3. Lab is a finite set of synchronization labels that contains the stutter label $\tau{\in}Lab$.

4. Edg is a finite set of transitions. Each transition e is a tuple $\langle l, a, \mu, l' \rangle$ where $l{\in}$ Loc and $l'{\in}Loc$ are a source and a target locations, $a{\in}Lab$ is a synchronization label, and μ is an action defined by a guarded set of assignments (updates), $\psi \Rightarrow \{\ x := \alpha_x \mid x{\in}Var\ \}$. where the guard ψ is a linear formula over the variables, and α_x is also a linear term.

5. Act is a mapping from locations in Loc to a set of activities to represent the flow dynamics. $Act(l)$ is a differential equation of the form $dP/dt = K$ where P is a real-valued variable, $P{\in}Var$. K is C^l for the case of energy consumption and 1 for a clock such as an inactivity timer ($dP/dt = 1$). C^l is an energy consumption rate at a location l.

6. Inv is a mapping from locations in Loc to invariants $Inv(l){\subseteq}V$. $Inv(l)$ is defined by a linear formula ϕ over Var.

A PCA generates a timed-sequence $\langle l_j, v_j, \tau_j \rangle$ where l_j, v_j and τ_j refer to a location, a valuation and a time point respectively. If P denotes a real-valued variable to account for the consumed energy, the total amount of energy is $\sum_i v_i(P)$, in which $v_i(P) = C^{l_i}{\times}(\tau_{i+1} - \tau_i)$.

4 Platform-Dependent Effects on Analysis

As one of the most significant platform-dependent aspects, we measured quantitative effects of power-saving processors on the energy consumption. Then, we discuss extensions of the PCA to take into account such effects.

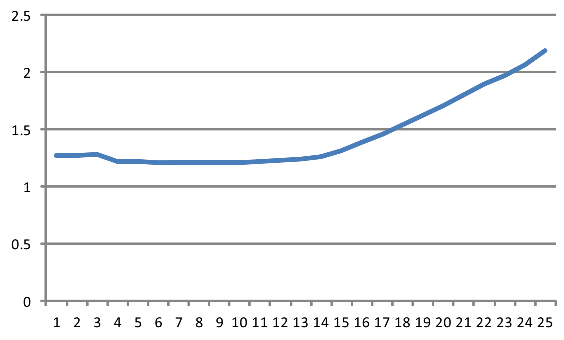
4.1 Power-Saving CPU

Mobile devices, such as Android smartphones, are equipped with ARM core processors [2] to allow the dynamic voltage-frequency scaling (DVFS) technique. The dynamic power is proportional to both the square of the operation voltage ($voltage^2$) and the frequency switched [7]. Changing voltage and frequency, however, has an impact on the execution time of an application program and thus on the energy consumption of hardware components used by the program.

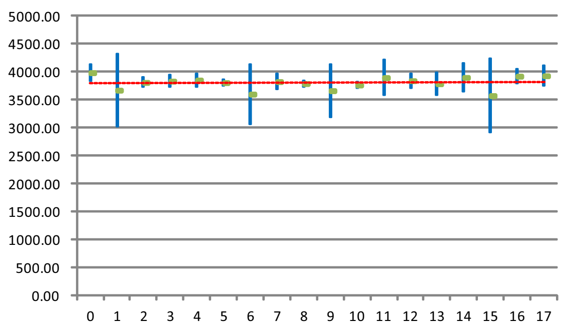
The DVFS governor of the Android framework is a variant of the ondemand governor [9] supported by the Linux kernel. The governor cooperates with the Linux process scheduler. Depending on the CPU load, the governor controls the supplied voltage and CPU operation frequency. When the load is small, low voltage and low frequency are chosen. As the CPU load becomes large, the governor adjusts the voltage and frequency to be higher than before.

The physical execution time of a program is longer in such a low power mode than the case with the full powered CPU. It is problematic in hard-realtime systems, but is not a big issue in smartphones or tablets. The major concerns here are the interactions with human users, which is considered soft-realtime. Reducing the energy consumption is a higher priority issue than the potentially long physical execution time.

While the energy consumed by CPU can be reduced, the effect of the DVFS governor may increase the total energy consumption especially when the application program has ebugs in it. If the execution time is longer, the Wi-Fi subsystem, for example, may consume more battery power because its energy consumption is dependent on the physical time. Therefore, the energy profile is affected indirectly by the DVFS governor.



(a) Average of Workload Threads



(b) A WiFi Client Application

Figure 3: Execution Times

Note that some processors such as Tegra3 [4] used in Nexus7 (2012 model) [3] are more complicated than what was mentioned above. It is because Tegra3 is a multi-core processor to contain five ARM cores in itself. The details is explained in Appendix A.

4.2 Summary of Measurement Results

4.2.1 Switching Frequencies

First, we conducted the measurements to observe the Tegra3 scenario of Appendix A. We used the experimental setup of Appendix B and confirmed that the changes in the number of operating CPU cores and the frequency certainly reproduced the scenario.

4.2.2 Workloads

Figure 3(a) plots the average execution time of the workload threads. The y-axis shows the time in seconds needed for the threads to complete a predefined number of iterations. The values are not significant, but the graph as a whole implies an interesting tendency.

From one to three threads, the values are almost constant. They drop a little to be a minimum at four threads; the program runs fastest at four workload threads. Then, the values increase gently from five to fourteen, which is followed by a continuing steep increase.

The execution time does not vary so much from one to fourteen threads and thus is considered almost constant. It shows that the DVFS governor accomplishes a good balance between the execution speed and energy consumption. The steep increase in the graph shows that the realtime response becomes bad. It is because the workload is larger than what the high-performance cores can support even at their highest frequency.

4.2.3 WiFi Client Application

Since it is always the case that the response time becomes worse as the workload increases, we decided to measure the execution time of a specific application in *moderate* workload environments. The numbers of the workload threads were changed from zero to seventeen. The result is shown in Figure 3(b), which is the execution time of this particular application program. In contrast, Figure 3(a) depicts the average behavior of many workload threads.

In the graph, the y-axis depicts the time measured in *msec*. The execution time is mostly constant around 3.8 seconds; the average ($\overline{\mu}$) is 3.8 seconds with the standard deviation ($\overline{\sigma}$) of 0.6 seconds. The total relative error is, thus, about 30% ($2\times\overline{\sigma}/\overline{\mu} = 1.2/3.8$). Such deviations may come from uncontrollable operating conditions to include changes in the WiFi signal strength. The variations in the execution time are considered as the statistical fluctuations.

4.3 Possible Extensions of PCA

As the DVFS governor has an impact on the physical execution time of application programs and thus may affect the energy consumption behavior, the dynamics of the PCA must include such effects.

4.3.1 Physical Execution Time

First, we review how the execution time of an application program is affected when the operation frequencies of processor are changed.

In processors, the number of clocks $\mathcal{F}$ in a time interval is dependent on the switching frequency f, and is, in the most general case, considered as a monotonically increasing linear function of f. We assume here that the function $\mathcal{F}(f)$ is linear with respect to frequencies f. If the maximum frequency of the processor (f_0) is chosen as a reference, the execution time (τ_0) of an application program is $A_p/\mathcal{F}(f_0)$ where A_p number of clocks are needed to finish the application. Then, $A_p = \mathcal{F}(f_0)\times\tau_0$. Let τ be the execution time of a processor at the frequency f. For this particular application program, A_p is a constant, and the relationships hold; $A_p = \mathcal{F}(f_0)\times\tau_0 = \mathcal{F}(f)\times\tau$. Then, $\tau = (\mathcal{F}(f_0)/\mathcal{F}(f))\times\tau_0$. Because of $\mathcal{F}(f_0) \geq \mathcal{F}(f)$ due the monotonicity of $\mathcal{F}(f)$, $\tau \geq \tau_0$ holds, which shows that physical execution time at a low frequency f is longer than the case with f_0.

4.3.2 Linear Hybrid Automaton

If the value of the function $\mathcal{F}$ is minimum at a frequency f_α, the relation $1 \leq \alpha$ holds for a constant α to satisfy that $\alpha = \mathcal{F}(f_0)/\mathcal{F}(f_\alpha)$. The f_α is the smallest since $\mathcal{F}$ is monotonically increasing. Then, dynamics of the PCA satisfies the inequalities,

$$C \leq dP/dt \leq \alpha{\times}C$$

where C represents a constant value to denote the rate of energy consumption at the maximum operation frequency f_0. dP/dt may take a large value by a factor of α when the frequency is the smallest. This variant of the PCA is strictly the same as the LHA [5]. The formal analysis is conducted in which the value of dP/dt is chosen non-deterministically from the interval.

In Tegra3, the high performance core operates at 1200MHz as the maximum (f_0) and 340MHz as the minimum (f_α) frequencies. If we simply assume that $\mathcal{F}(f)$ is proportional to f (written $\mathcal{F}(f){\propto}f$), $\alpha = 1200/340 = 3.5$. We must, however, consider measurement results of the execution time of programs. Figure 3(a) shows that the ratio α is about 2, which is taken from the ratio of the worst execution time to the best during the workloads of one to twenty five threads. Although this measured ratio is smaller than 3.5, a larger value must be chosen so that the analysis method does not miss any possible variations even when the workload is large. Therefore, if formal analyses, such as the reachability, are conducted for this PCA extension, the results will be over-approximations. There are high chances to produce spurious alarms, and thus the model may not be appropriate.

4.3.3 Probabilistic Hybrid Automaton

The function $\mathcal{F}$ can be studied a bit in detail for the case of the control method used in Variable SMP of NVIDIA [4] (see Appendix A). A simple relationship is assumed so that $\mathcal{F}(f) = f{\times}M{\times}u$, where M is the number of operating cores, and u ($0{<}u{\leq}1$) is a kind of utilization factor for application programs to run on the processors. The value of $\mathcal{F}$ is maximum, denoted by $\mathcal{F}_0$, when all the processor cores are operating at the maximum frequency f_0 and the utilization is 100%. Then, it becomes $\mathcal{F}_0(f_0)/\mathcal{F}(f) \propto 1/(f{\times}M{\times}u)$.

The utilization factor and operation frequencies vary and thus are not known beforehand. They are supposed to follow a certain probabilistic distribution. A possible extension of the PCA may show stochastic behavior due to such probabilistic distributions.

A probabilistic variable R_i is introduced to follow a distribution function $g(R)$, namely $R_i \sim g(R)$. A function $r(R)$ to take probabilistic values is introduced that $1 \leq r$ and $r(R) \propto 1/(f{\times}M{\times}u)$.

$$dP/dt \;=\; r(R_i){\times}C$$

The PCA is now a subclass of LHA, but shows probabilistic behavior, namely a probabilistic LHA. Statistical model-checking methods [13] will be employed for the formal analysis of this extension of the PCA. The probabilistic distribution function $g(R)$ may be a Poisson distribution in which the relative error ($\overline{\sigma}/\overline{\mu}$) is around 15% if we follow the statistical behavior that Figure 3(b) shows.

This probabilistic extension of the PCA is considered more faithful than the LHA-equivalent. The probabilistic PCA includes uncontrollable effects on the energy consumption behavior of target application, which are represented in the dynamics.

5 Discussions

Figure 3 indicates that the execution time varies about 30% because of uncontrollable operation conditions. Runtime profilers may always have such an amount of statistical errors in the measurement. If the extra energy consumption due to ebugs results in an increase of less than 30%, the profiler does not distinguish the outcome of the ebugs from the measurement errors. As Figure 3(a) suggests, the variations will be larger as the workloads become higher. It implies that the effects of ebugs are hidden in the large variations due to the measurement errors. Model-based methods are needed that do not rely on monitoring program executions.

As Figure 3 (b) suggests, the execution time of the target application program is almost constant from zero to seventeen worker threads. If we consider that the formal analysis of the energy consumption behavior is conducted under these moderate workloads, it need not take into account the impact of the DVFS processors. Therefore, the basic PCA model with its dynamics $dP/dt = C^l$ (Section 3) can be taken as an appropriate model for the application behavior.

We now consider how the two PCA models are linked. The basic PCA extracts the energy consumption behavior of application programs without taking into account of any effects from the execution environments. The analysis results with this basic PCA provide a piece of *qualitative* information to be used in finding ebugs. They are not what can be used for predicting a *quantitative* amount of the energy consumption.

The probabilistic PCA introduces a probabilistic component taking a form of $r(R)$. The model-based method with the probabilistic extension is similar to the profiler-based method in the sense that they calculate the amount of the energy consumption under *uncontrollable* operating conditions. Therefore, their numerical results could be compared.

As for the relationship between the basic PCA and probabilistic PCA, we elaborate the basic model into a probabilistic one. This elaboration may be compared with the notion of *usual* refinement, where non-deterministic behavior is refined into a concrete deterministic one.

Last, for the case of non-functional concerns such as energy consumption, we summarize our conjecture that behavioral models are essential and that the deterministic behavior is elaborated into the probabilistic one so as to compare the model (model-based) and implementation (profiler-based).

6 Related Work

A. Pathak et al recognized the importance of eliminating energy bugs in smartphones, which they called *ebugs* [11]. They also proposed to use state-transition systems [10] for modeling the asynchronicity of the energy consumption, where the model was presented informally. Based on the state tracing techniques, Eprof [12] is an energy profiler to monitor the program execution at runtime to detect potential ebugs. ADEL [14] is a runtime profiler to employ a taint-tracking method to detect asynchronous energy leaks.

S. Nakajima [8] proposed the idea of the PCA. The PCA model was inspired by [10], but was the first formal model to make explicit the relationship with LHA. This paper, based on the basic PCA, studied the impacts of the DVFS governor by using the measurement experiments, and pointed out the importance of the probabilistic extension of the PCA.

7 Conclusion

Since energy consumption is a physical phenomenon, a *right* abstract model in model-based analysis methods must take into account of experimental measurement results. Although the model-based method looks at the problem in a viewpoint different from the profiler-based one, a probabilistic model may link between them. The probabilistic model was introduced as a consequence of analyzing the measurement results.

References

[1] Android. `http://developer.android.com`.

[2] ARM Limited (2005), *IEM Software, Technical Overview* .

[3] ASUS (2012) Nexus7.

[4] NVIDIA (2011), *Variable SMP – A Multi-Core CPU Architecture for Low Power and High Performanc3*.

[5] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine (1995), *The Algorithmic Analysis of Hybrid Systems*, Theor, Comp. Sci., No.138, pp.3-24.

[6] P. Bezzera, L. Araujo, G. Ribeiro, A. Neto, A. Silva-Filho, C. Siebra, F.Q.B. da Silva, A. Santos, A. Mascaro, and P. Costa (2013), *Dynamic Frequency Scaling on Android Platforms for Energy Consumption Reduction*, In Proc. PM2HW2N'13, pp.189-196.

[7] J.L. Hennessy and D.A. Patterson (2011), *Computer Architecture : A Quantitative Approach (5ed.)*, Morgan Kaufmann.

[8] S. Nakajima (2013), *Model-based Power Consumption Analysis of Smartphone Applications*, In Proc. ACES-MB'13.

[9] V. Palipadi and A. Starikovskiy (2006), *The Ondemand Governor*, In Proc. Linux Symp. 2006.

[10] A. Pathak, Y.C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang (2011), *Fine-Grained Power Modeling for Smartphones Using System Call Tracing*, In Proc. EuroSys'11.

[11] A. Pathak, Y.C. Hu, and M. Zhang (2011), *Bootstrapping Energy Debugging on Smartphones: A First Look at Energy Bugs in Mobile Devices*, In Proc. Hotnets'11.

[12] A. Pathak, Y.C. Hu, and M. Zhang (2012), Fine Grained Energy Accoutning on Smartphones with Eprof: Where is the energy spent inside my app?, In Proc. EuroSys'12.

[13] H.L.S. Younes, M. Kwiatkowska, G. Norman, and D. Parker (2006), Numerical vs. Statistical Probablistic Model Checking, J. STTT, 8(3), pp.216-228.

[14] L. Zhang, M.S. Gordon, R.P. Dick, Z.M. Mao, P. Dinda, and L. Yang (2012), *ADEL : An Automatic Detector of Engery Leaks for Smartphone Applications*, In Proc. CODES+ISSS'12.

A Energy Management in Nexus7

Nexus7 (2012 model) [3] employs NVIDIA Tegra3. It is a multi-core processor, and takes a form of "4+1" to have four high-performance cores and a single low power core, all of which have an identical ARM architecture. ARM core comes with Intelligent Energy Manager (IEM) [2], a low-level driver program to change the frequencies of the core dynamically. The IEM is provided for Linux and used by the power management driver (`CPUFreq`) and DVFS governor. Linux, furthermore, has a driver program (`CPUHotplug`) to manage the multi-cores by dynamically changing the number of cores to operate.

Tegra3 adapts the Variable SMP architecture [4], and assumes the following usage scenario. It manages the dynamic power consumption by changing both the frequencies and the voltage power to drive the ARM core circuits by adjusting the number of cores to operate.

1. The low power core is used when the CPU load is low.
2. `CPUFreq`, under control of a DVFS governor, increases dynamic frequencies of the running ARM core as the CPU load becomes large.
3. When the frequency reaches a pre-defined threshold, `CPUHotplug` chooses one of the high-performance cores to operate at a pre-determined start-up frequency. The low power core stops at this moment.
4. The frequency is increased in the same manner as in 2) to catch up the increase in the CPU load.
5. When the frequency is increased to reach a pre-defined threshold, `CPUHotplug` chooses another high-performance core to operate. `CPUFreq` makes all the chosen cores to operate at a same start-up frequency.
6. The frequency is increased in the same manner as in 2). Note that all the cores operate at the same frequency.
7. If the CPU load further becomes high, `CPUFreq` and `CPUHotplug` controls Tegra3 as in 4) and 5).
8. If the CPU load becomes low, `CPUFreq` lowers the dynamic frequency. Furthermore, `CPUHotplug` stops a core when the frequency is decreased to reach the low threshold.

B Experimental Setup

B.1 Basic Approach

The experiment is conducted to use Nexus7 (2012 model), and measures how the operation frequencies and number of running cores are changed when the workload is increased monotonically. First, we confirm that such changes follow the scenario for Tegra3 explained in Appendix A. Second, we study how the changes in workload have impacts on the execution time of programs. In the measurements, the tablet is set in the *in-flight* mode, and the applications or services are stopped as much as possible.

The experiment method is solely based on the Android/Linux features that are transparent to application programs to make the measurement method portable. We do not employ any method to use external hardware nor to introduce modifications in the kernel codes.

B.2 Switching Frequencies

The dynamic switching frequencies of the processor cores are obtained from Linux pseudo files. The files record the percentage of a particular frequency chosen in a particular duration. The values can be obtained by monitoring periodically the files `cpu?/cpufreq/stats/time_in_state` under the folder `/sys/services/systems/cpu/`. The file under `cpu0` records either the low power core or a high performance core chosen first. The files under `cpu{1-3}` exist only when the corresponding core is in operation; the file does not exist when the core is not operating.

B.3 Workloads

A simple application program is developed for affecting the CPU load. It is multi-threaded and the load changes are made easy by increasing the number of threads. The program iterates its body of computation, which makes it easy to know how much computation is done.

B.4 WiFi Client

An application program to use WiFi communication is also developed to control the WiFi behavior using `WiFiLock` methods. The execution history of such method calls are recorded by the Android; its data format is defined in class `android.os.BatteryStats`.

Feasibility Study in the Use of Contract-Based Approaches to Deal with Safety-Related Properties in CPS

Daniela Cancila
CEA, LIST, CEA Saclay - F91191 Gif-sur-Yvette Cedex; email: daniela.cancila@cea.fr
Elie Soubiran
Technological Research Institute SystemX - Alstom Transport; email: elie.soubiran@{irt-systemx.fr,transport.alstom.com}
Roberto Passerone
Dipartimanto di Ingegneria e Scienza dell'Informazione - University of Trento, Italy; email: roberto.passerone@unitn.it

Abstract

This work concerns a feasibility study on the use of contract-based approaches as a means of reasoning and understanding a cyber-physical system (CPS) which should meet safety properties. We show the problems, the analysis methodology and the results on a railway industrial system case study. Our results suggest that contract-based design provides a rigorous approach for reasoning at the interaction of safety-related properties in CPS.

Keywords: contract-based approach, CPS, Railway system, mixed-critical and safety-related properties.

1 Introduction

In the last decade, Cyber-Physical Systems (CPS) have assumed an increasingly significant role in a number of disciplines, especially in Computer Science, and form one of the cornerstones of the study of dynamical and heterogeneous systems. CPS combine signals from physical components with (embedded) software components and integrated circuits.

Historically, the term 'cyber-physical systems' was first introduced by H. Gill to broadly capture a similar meaning of the term 'cyberspace' and 'cybernetics' [1]. Since then, the term CPS has been widely adopted by the scientific community and, today, it appears as one of the main topics of the European projects (e.g., H2020, EIT ICT Labs).

Contract-based approaches are considered as a promising means to deal with CPS [2,3,4,5,6,7,8]. A contract is a pair (assumption, guarantee), where the guarantee specifies the functionality provided by a component to the environment; and the assumption sets forth the conditions required from the environment in order for the component to accomplish its guarantee [5]. The contracts, which are specifications on both physical and computational components, help us identify precisely the conditions for a correct interaction.

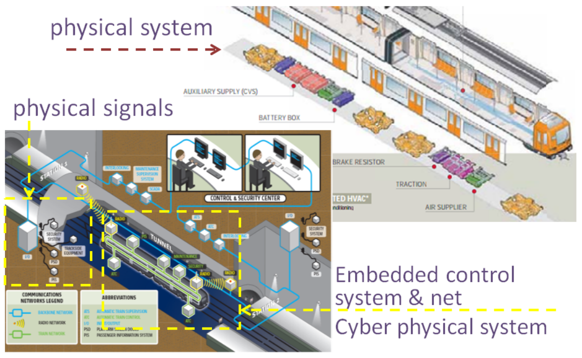


Figure 1: Image extracted from 'Metropolis And Metro Train Solution' by Alstom [11]

This position paper arises from the FSF project (*Fiabilité et Surêté de Fonctionnement* Reliability and Safety) [9]. The bulk of the FSF project deals with safety-related properties of a railway system that involves components, which have an inherent different nature and, to complicate the scenario further, combine different safety integrity levels (SIL) [10]. This work is a feasibility and preliminary study that explores a contract-based approach to deal with a seamless guarantee of safety-related properties from CPS design to execution platform. We feel that this approach can provide a simple, but firm, foundation to a rigorous approach for reasoning about the interaction of safety-related properties in CPS.

2 Case Study

Figure 1 shows both the mechanical part and the cybernetic part (i.e., command, control and supervision) of a railway system. A first command and control loop takes place within train units, where embedded software subsystems ensure automatic train driving and protection. These subsystems are mostly safety critical and shall furthermore consider real-time constraints. A second loop takes place at the line level, and is concerned with line supervision (train-traffic, timetable, etc.) and focuses on operational performance.

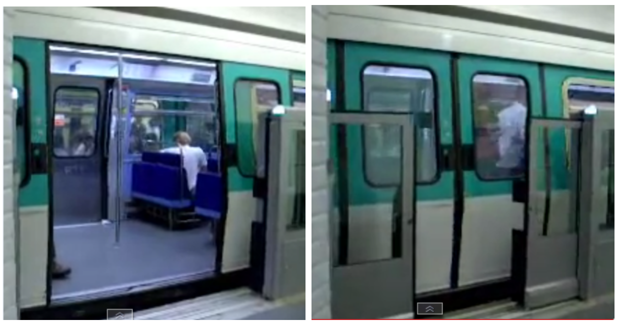


Figure 2: On the right, automatic opened doors, on the left, the platform doors are automatically closing (images extracted from youtube)

The case study considered in this paper is in the scope of the Communication Based Train Control (CBTC) system [12], and considers more precisely a subset of the Automatic Train Control subsystem (ATC). The associated operational scenario is the following: a train stops at a station that is equipped with a physical barrier and automatic doors, whose purpose is to protect passengers from the moving train (see Figure 2). In order to be able to operate train and platform doors, the doors of the train and the doors of the platform need to be aligned. At that point, both of them are automatically opened - thus allowing the passengers to get on and off the train. We will refer to this phase with the technical term *passenger exchange* in the rest of the paper. Finally, the train is authorized to move on if and only if both platform and train doors are closed.

The function *passenger exchange* is an important functionality of the CBTC, and this case study is obviously representative of CPS. Indeed, it integrates not only computational and physical processes with feedback loops, but also the human factor. This function takes control of platform and train doors when the train is safely docked at a station; then it organizes the exchange of passengers (e.g. manage train and station doors opening/closing and doors blocking by passengers) while protecting them from any untimely train movement or non-aligned doors opening. It finally gives the departure authorization when all safety conditions are met.

In this CPS we find different levels that co-exist, each of them with its own needs, requirements, guarantees. For example (list non-exhaustive):

- the door presence sensor, which ensures that no passenger is blocked between doors;
- acoustic and visual signalization, placed both on the platform and train side, which warn about the closing and opening doors.

The operational phase linked to this case study is critical since doors are open and passenger can move freely between the train and the station. Thus, it is relevant to focus the study on safety related properties that may be expressed and refined through contract-based analysis. To do so, we propose to start from identified hazards that cause accidents and/or near-miss accidents, then to establish contracts between the system components to define the necessary conditions that ensure safety, and then to refine those contracts down to software components and their associated computation unit. Beyond characterizing functional behaviours that would ensure safety invariant, the goal of contracts here will also be in a near future to support non-functional properties refinement and analysis with for instance SIL allocation, failure rate and so on.

3 Methodology

The CPS is initially modeled in SysML in the Papyrus tool - thus providing a holistic view of the whole system. For the sake of industrial adherence and industrial transfer of our work, we exploit the Alstom methodology to develop the model [13, 14]. The next paragraph reports the main principles of the quoted methodology, freely extracted from the Alstom documents [13].

In the last years, Alstom has developed the *Advanced System Architect Program* methodology, known as ASAP methodology, to increase quality of the system specification. In the methodology, textual requirements are initially deployed on model elements and are then further specified and refined. The modelling approach is threefold:

- operational vision, which deals with objectives and missions (why);
- functional vision, which concerns the strategy to perform missions (what);
- constructional vision, which addresses elements required to perform functions (how).

Alstom adopts the standard SysML language to implement the ASAP methodology. This latter has been tested on the Rolling Stock railway system, from Customer requirements/needs to product solution [13]. Some interesting industrial feedback on the use of SysML is provided by M. Ferrogalini and J. Le Bastard [14].

As firstly introduced, the ASAP methodology allows us to deal with physical signals, business needs, system specification and requirements. Therefore, we strategically adopt the ASAP methodology to specify the SysML model at an early stage of the development phase of our use case. When we refine the model further, however, we should be able to capture some details and then a component-based system engineering (CBSE) methodology seems to fit this scope better. In that context, a functional architecture is designed within the functional viewpoint, then resulting functions are allocated to components which belong to the constructional viewpoint. Following the SysML language primitives, components are represented by blocks, data by types and data transmission by port and connectors.

Our work strengthen the ASAP and the CBSE methodologies with a contract-based design approach.

3.1 Contract Specification

We adopt a textual format to introduce contracts at the CPS level. This approach fits better with high-level requirements, which are usually expressed in natural language. Our notion of contracts is based on previous work [5, 6, 7]. To the best of our knowledge, the ASSERT FP6 European project was the first to structurally establish the deployment of contracts on UML ports (and its profiles such as SysML or MARTE) [5, 15]. After that, several European research projects have widely adopted the relationship contracts - UML (and profiles) ports and successfully converged on it (see, for instance, the CHESS Artemis project [16]).

An intriguing use of contracts as a means to establish a firm relationship between software and control in CPS design has been recently introduced in the literature by Derler et al. [7]. There, functionality and timing are correlated in each of four types of contract to design effective control loops. This approach leads precision as well as abstraction - thus being easily applied to our use case.

Moreover, contracts are on one hand a means to prove correctness of heterogeneous components (through the notion of composability [17]), and, on the other hand, to prove the faithful refinement between two abstraction levels of a design [6]. In order to ensure continuous and automatic verification throughout the specification, the design and implementation phases, we are forced to eventually specify contracts by a formal, and non ambiguous, language. At this step of the development, we envisage adopting a similar language to that introduced in the literature [18] and, more recently, adopted by the Autosar consortium [19].

When we refine the system further, we follow the Platform-Based Design approach (PBD) [20, 21, 22]. This approach has been widely adopted by the scientific and industrial community, albeit not without difficulties and following several approaches [23]. Nonetheless, PBD allows us to introduce a common semantic domain between different abstraction levels as well as different views of a design, which help to maintain a consistent view of the system.

3.2 HMI and contract visualization

From a visualization point of view, 2D or 3D representations could help the designers have a better grasp of their systems. More in particular, a 3D representation could help us (and final costumers) reason about the physical aspects of CPS. It would provide a mean to simulate the CPS regarding different operational scenario and their respective impact on contracts. However, when we deal with automatic verification, we consider SysML UML supporting 2D tools, such as Papyrus, Obeo Designer, IBM or Atego, which are easily customizable.

3.3 Safety and Certification

Safety issues have a prominent role, especially in those CPS which ought to entail a certification process. This is exactly the case of some functionality and mechanical components of our use case. For example the *Passenger exchange* functionality and the mechanical signalling components involve the highest safety integrity level.

Each company has its own *savoir-faire* to identify and analyze the safety properties. Usually, Safety engineer teams identify and deeply study accident scenarios and identify barriers that mitigate the risk to an acceptable level. For instance, in the case study, an accident could result from a train that departs when the door are not yet properly closed. A functional barrier is then identified and provides a safe departure authorization to the train.

The performed analysis should be compliant to the related safety norms and validated by an independent certification entity. In many cases, the results of that analysis take the form of requirements, which identify safety barriers, such as preventive and palliative ones (non-exhaustive list).

Safety requirements should be adequately taken into consideration in all development phases of the system: from the specification to maintenance. As a result, their traceability is a key component of methodologies oriented towards the development of critical systems.

4 Application to the Case Study

In many cases, current industrial processes provide a list of requirements in a textual format. Not only are these latter exploited/improved during all development phases, but they are also used during the certification/qualification phase: the validator checks that (textual) code is compliant with all (textual) requirements.

The companies, which base industrial systems specification and analysis on component-based approaches, often adopt a bidirectional tool from textual requirements space to design modeling space. Then, they deploy requirements to model elements.

Like the industrial practice, in our approach a requirement is initially imported by a textual document.

[Req.] *The Passenger exchange train control function shall determine which train and platform doors are enabled for opening, based on vital localization (with regards to the track platforms) and kinematic conditions.*

The quoted requirement addresses the train control functionality that allows the system to automatically open/close both the train and platform doors, under certain conditions (e.g., vital localization, kinematic conditions).

Then, the requirement is further specified by adopting a contract-based approach. We firstly identify the assumptions from the original text:

a_1 Valid and defined kinematic conditions;

a_2 Valid and defined vital train localization;

a_3 List of platforms described by their position on track, and the position of each platform door.

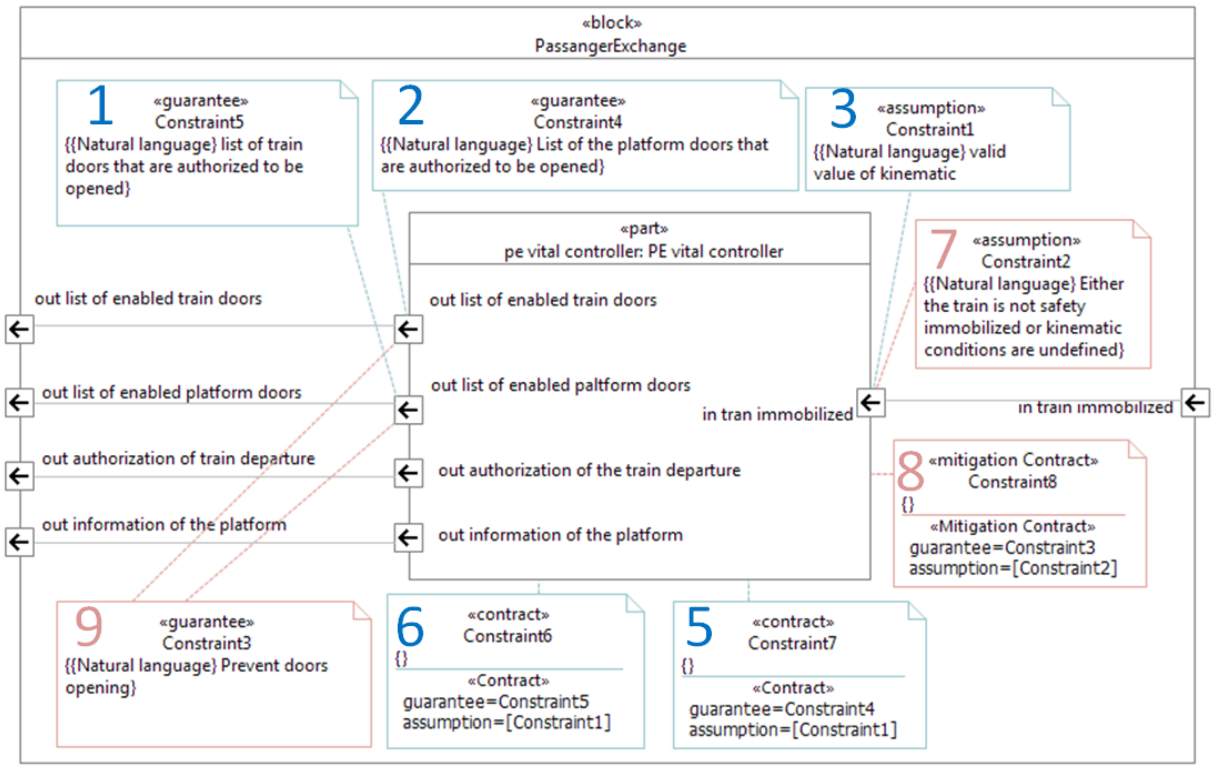

«block»
PassangerExchange
1 «guarantee» Constraint5
{{Natural language} list of train doors that are authorized to be opened}
2 «guarantee» Constraint4
{{Natural language} List of the platform doors that are authorized to be opened}
3 «assumption» Constraint1
{{Natural language} valid value of kinematic
«part»
pe vital controller: PE vital controller
7 «assumption» Constraint2
{{Natural language} Either the train is not safety immobilized or kinematic conditions are undefined}
out list of enabled train doors
out list of enabled train doors
out list of enabled platform doors
out list of enabled paltform doors
in tran immobilized
in train immobilized
out authorization of train departure
out authorization of the train departure
8 «mitigation Contract» Constraint8
{}
«Mitigation Contract»
guarantee=Constraint3
assumption=[Constraint2]
out information of the platform
out information of the platform
9 «guarantee» Constraint3
{{Natural language} Prevent doors opening}
6 «contract» Constraint6
{}
«Contract»
guarantee=Constraint5
assumption=[Constraint1]
5 «contract» Constraint7
{}
«Contract»
guarantee=Constraint4
assumption=[Constraint1]

Figure 3: Contract-based approach to Model-Based system engineering

Moreover, we identify the guarantees. For the sake of brevity, we intentionally combine functional with non-functional properties in the guarantees specification. However, to properly deal with non-functional properties, two types of contracts and views are needed. We omit further details because they are out of the scope of this work.

In Guarantee g_1 and Guarantee g_2, timing specifies the maximum value of timing for which a datum remains valid. After the deadline, validity of the datum is no longer ensured; for safety reasons, it should re-calculated and required again.

g_1 Determine which **train doors** are enabled for opening. The validity duration of this value is set to 1200 msec. Undefined values shall be interpreted as not enabled;

g_2 Determine which **platform doors** are enabled for opening. The validity duration of this value is set to 1200 msec. Undefined values shall be interpreted as not enabled.

Finally, we introduced two contracts:

$$C_1 = \{a_1, a_2, a_4; g_1\} \text{ and } C_2 = \{a_1, a_2, a_3; g_2\}.$$

We model contracts in a SysML environment as follows. We deploy guarantees and assumptions to the ports of a component and contracts to the element (Figure 3). Moreover, we identify the 'constraint' UML model element to specify guarantee, assumption and contracts. Our choice is founded on two principles: to be able to deploy more than one guarantee (resp. assumption) on the same model element, and to easily access them, using the graphical facilities of the Papyrus tool.

We specify the remaining requirements via a contract-based design. We discover that some requirements are not directly refined from the top-level requirement; instead, they derive from the safety analysis (Preliminary Safety Analysis and System Hazard Analysis) and they are introduced to mitigate, or avoid, possible accidents. We trace them with suitable contracts.

Figure 3 traces two types of contracts:

- Functional contracts (graphically the blue boxes, which are highlighted with numbers from 1 to 6), which describe the functional behavior; and
- Safety contract (graphically the red boxes, which are highlighted with numbers from 7 to 9) which represents safety barriers.

Our investigation shows that functional contracts are directly derived from the top-level requirement [Req], previously quoted. However, this is not the case of safety contracts. Although this latter specifies [Req] further, it is not directly derived from [Req]. It refines a safety requirement, which has been firstly identified, secondly studied and analyzed, and, then, required to be introduced in the design specification, by the safety engineer teams to ensure the safety integrity level entailed by the CPS.

The (red and Number 8) contract has a means to highlight traceability of safety requirements, which are previously captured by the safety engineers teams during the Hazard Analysis at the early stage of the system development.

At the meta-modeling level, we then introduce Stereotype 'MitigationContrats' that has the primary role to trace the link

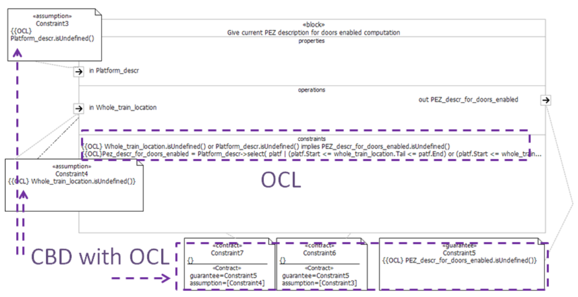

«assumption»
Constraint3
{{OCL}
Platform_descr.isUndefined()
«block»
Give current PEZ description for doors enabled computation
properties
in Platform_descr
operations
in Whole_train_location
out PEZ_descr_for_doors_enabled
constraints
{{OCL} Whole_train_location.isUndefined() or Platform_descr.isUndefined() implies PEZ_descr_for_doors_enabled.isUndefined()
{{OCL}Pez_descr_for_doors_enabled = Platform_descr->select(platf | (platf.Start <= whole_train_location.Tail <= patf.End) or (platf.Start <= whole_train...
OCL
«assumption»
Constraint4
{{OCL} Whole_train_location.isUndefined()}
CBD with OCL
«contract»
Constraint7
{}
«Contract»
guarantee=Constraint5
assumption=[Constraint4]
«contract»
Constraint6
{}
«Contract»
guarantee=Constraint5
assumption=[Constraint3]
«guarantee»
Constraint5
{{OCL} PEZ_descr_for_doors_enabled.isUndefined()}

Figure 4: Contract-based approach to Model-Based system engineering

between a contract at design space and the original specification at safety space.

Figure 4 shows a comparison between requirements specified via contract-based approaches, and requirements specified with a textual flat language. We intentionally adopt the same formal language: the international OMG standard 'Object Constraint Language' (OCL) [24], which is compliant with SysML and hence the two standards can be easily applied together to the same model. OCL is a formal language that allows engineers to specify requirements or more in general constraints, thanks to the help of a formal syntax, in a model previously specified (for example by UML, SysML, MARTE).

Figure 4 shows two contracts: they have the same guarantee, but differ from the assumptions. The assumptions and guarantee are clearly deployed on the related model elements and are correlated via a contract.

The block includes an OCL constraint, specified in the usual manner. The constraint has the following form $A \vee B \rightarrow C$, where A and B correspond to the previous assumptions and C to the guarantee. However, such a flat formulation does not clearly highlight the association between the atomic formula (A, B or C) and the model element; the only way we have to recognize such a correspondence is by the name (for example, Whole_train_location.isUndefined() in the formula corresponds to the Port with name Whole_train_location).

An advantage in the use of contract-based approaches is to structure the link between an OCL atomic formula and the corresponding model element.

4.1 Preliminary Feedback

During this work, we have been able to compare CBSE with the textual requirements approach and CBSE with the textual contracts approach. Even if the expressive power remains equivalent, contracts have the advantage to drive the component breakdown structure analysis and design by facilitating the allocation and refinement of functional and safety behaviours on sub-components. It seems also a promising mean for structuring verification and validation activities. Finally, thanks to their inherent ability for traceability, contracts are good candidates to strengthen a development process compliant with CENELEC norms.

5 Conclusion and On-Going Work

In this position paper, we introduce the overall view we pursue to deal with seamless guarantee of safety-related properties from CPS design to execution platform in the FSF project [9]. The vision outlined exploits contracts as a means to identify precisely the conditions for a correct interaction of components as well as to specify which assumption a functional level (code) should require to a hardware level to ensure the acceptable threshold of SIL. Although our work is at an early stage of development, we feel that this approach can provide a simple, but firm, foundation to a rigorous approach for reasoning on the interaction of safety-related properties in CPS.

6 Acknowledgment

This research work has been carried out under the leadership of the Technological Research Institute SystemX [25], and therefore granted with public funds within the scope of the French Program *'Investissements d'Avenir'*. This work was also supported in part by the CyPhERS FP7 project, grant agreement no. 611430.

References

[1] C. Ptolemaeus (2014), ed., *System Design, Modeling, and Simulation using Ptolemy II*, Ptolemy.org.

[2] L. de Alfaro and T. A. Henzinger (2001), *Interface automata*, in Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering, pp. 109–120, ACM Press.

[3] L. Benvenuti, A. Ferrari, L. Mangeruca, E. Mazzi, R. Passerone, and C. Sofronis (2008), *A contract-based formalism for the specification of heterogeneous systems*, in Proceedings of the Forum on Specification, Verification and Design Languages, FDL08, (Stuttgart, Germany), pp. 142–147, September 23–25.

[4] R. Passerone, I. B. Hafaiedh, S. Graf, A. Benveniste, D. Cancila, A. Cuccuru, S. Gérard, F. Terrier, W. Damm, A. Ferrari, L. Mangeruca, B. Josko, T. Peikenkamp, and A. Sangiovanni-Vincentelli (2009), *Metamodels in Europe: Languages, tools, and applications*, IEEE Design and Test of Computers, vol. 26, pp. 38–53.

[5] D. Cancila, R. Passerone, T. Vardanega, and M. Panunzio (2010), *Toward Correctness in the Specification and Handling of Non-Functional Attributes of High-Integrity Real-Time Embedded Systems*, in IEEE Transactions on Industrial Informatics.

[6] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone (2012), *Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems*, in European Journal of Control, vol. 3, pp. 217–238.

[7] P. Derler, E. A. Lee, M. Torngren, and S. Tripakis (2013), *Cyber-physical system design contracts*, in International Conference on Cyber-Physical Systems (ICCPS 2013), (Philadelphia , USA).

[8] CyPhERS FP7 Project, *Cyber-Physical European Roadmap and Strategy*. http://cyphers.eu/.

[9] FSF IRT SystemX Project, *Fiabilité et Surêté de Fonctionnement (Reliability and Safety)*. http://www.irt-systemx.fr/wp-content/uploads/2013/03/FiabiliteetSuretedeFonctionnement.pdf.

[10] CENELEC (2012), *Railway applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 2: Systems approach to safety*, CENELEC standard http://www.cenelec.eu/.

[11] ALSTOM, *Metropolis And Metro Train Solution*, http://www.alstom.com/.

[12] IEEE (1999), *IEEE Standard for Communication Based Train Control Performance Requirements and Functional Requirements*, IEEE standard. http://standards.ieee.org/.

[13] ALSTOM, *Alstom ASAP methodology: Advanced System Architect Program* OMG http://www.omgwiki.org/MBSE/doku.php?id=mbse:alstomasap.

[14] Marco Ferrogalini, Jean Le Bastard, *Return of experience on the implementation of the System Engineering approach in Alstom* OMG http://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:rex_on_se_approach_implementation_in_alstom.pdf.

[15] ASSERT FP6 Project, *Automated proof-based System and Software Engineering for Real-Time systems project*.http://www.assert-project.net.

[16] CHESS Project, *Composition with Guarantees for High-Integrity Embedded Software Components Assembly*. http://www.chess-project.org.

[17] J. Sifakis (2005), *Embedded Systems - Challenges and Work Directions*, in Principles of Distributed Systems, (LNCS, ed.), vol. 3544.

[18] W. Damm, H. Hungar, B. Josko, T. Peikenkamp, and I. Stierand (2011), *Using Contract-based Component Specifications for Virtual Integration Testing and Architecture Design*, in Proceedings of DATE conference, pp. 109–120.

[19] AUTOSAR, *Automotive Open System Architecture*. www.autosar.org.

[20] A. Pinto, A. Bonivento, A. L. Sangiovanni-Vincentelli, R. Passerone, and M. Sgroi (2006), *System level design paradigms: Platform-based design and communication synthesis*, ACM Transactions on Design Automation of Electronic Systems, vol. 11, pp. 537–563.

[21] A. L. Sangiovanni-Vincentelli (2007), *Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design*, Proceedings of the IEEE, vol. 95, pp. 467–506.

[22] A. Davare, D. Densmore, L. Guo, R. Passerone, A. L. Sangiovanni-Vincentelli, A. Simalatsar, and Q. Zhu (2013), METROII*: A design environment for cyber-physical systems*, ACM Transactions on Embedded Computing Systems, vol. 12, pp. 49:1–49:31.

[23] D. Densmore, R. Passerone, and A. L. Sangiovanni-Vincentelli (2006), *A platform-based taxonomy for ESL design*, IEEE Design and Test of Computers, vol. 23, pp. 359–374.

[24] OMG, *Object Constraint Language (OCL)*. OMG standard.http://www.omg.org/spec/OCL/.

[25] IRT, *Institut de Recherche Technologique (Technological Research Institute)*. http://www.irt-systemx.fr/.

RoundTable on “Challenges and New Approaches for Dependable and Cyber-Physical System Engineering (De-CPS)”, Ada-Europe 2014

Daniela Cancila* ***Jean-Louis Gerstenmayer***
CEA, LIST, CEA Saclay - F91191 Gif-sur-Yvette Cedex - France; email: {firstname.name}@cea.fr
Charles Robinson ***Laurent Rioux****
THALES R&T, 1 Av. Augustin Fresnel 91767 Palaiseau Cedex - France; email: {firstname.name}@thalesgroup.com

1 Main Axes

In this article a synthesis is provided of the discussion that took place following the presentations from invited speakers. The topics presented ranged across modeling of critical cyber-physical systems, contracts-based engineering, co-engineering methods, standardization and formal methods. There were participants at the workshop from academia and different sectors of industry such as automotive, railway and SMEs. The international presence from Japan provided interesting perspectives.

The roundtable, led by J.L. Gerstenmayer, highlighted four main challenges (from Section 1.1 to Section 1.4) .

1.1 Handling the impact in the separation of functional and non-functional attributes to meet correctness-by-construction methodology

The separation of functional and non-functional attributes has been strongly promoted by the academic and industrial communities to improve reuse of components. In many industrial systems, this separation is a refined practice for the engineering of industrial systems. The ever increasing complexity of systems and materials, which require more and more performance, suggests first to deploy functionality with different levels of heterogeneity (for example in safety levels or temporal attributes) on the same material, and, secondly, to exploit material mechanisms to ensure safety and security properties. The control of a system’s behavior to activate the appropriate safety measures/mechanisms in the event of errors occurring involves the verification and the control of non-functional attributes, including the real-time-related ones. It is a particular necessity for cyber-physical systems which involve critical aspects.

The scientific and industrial community agrees with the following sentence: the more we anticipate the verification that a system meets its temporal constraints, the more we reduce the risk to find errors late in the design and development process. In other words, the control of real-time properties is a cornerstone to achieve the correctness-by-construction approach to design.

During the workshop, some represented industries promoted and highlighted the *critical* importance to specify real-time properties early in the design phase. However, despite compulsory acceptability thresholds for non-functional properties, at the early stage of the design process in many industrial systems, the specification of a system deals with functional properties only. Preliminary analysis of safety concerns and the introduction of safety mechanisms can reduce the risk of an accident. Although real-time parameters play an important role in correctness-by-construction methodologies, they are often specified and analyzed later in the design phase. The main reason is that they are related to the adopted material. Is it realistic to expect changes in existing specification process for industrial systems, that work well, by highlighting real-time attributes at an early stage in the development phase? How much does this operation cost in terms of human resources, effort and money?

However, if the material changes, by virtue of the use of new technologies, the real-time and effective behavior of the material has a direct impact on the safety analysis and certification of the system.

This complex, and often contradictory industrial context (cost vs novelty) provides the core of the workshop discussions of the this first issue. Without the ambition to settling the issue during the workshop-day, we agreed with the importance to address the following challenges and directions in the comming years: to devote effort in structuring the relationship between software and material requirements, between software and material technical mechanisms to ensure safety properties, and finally between software and material teams, which usually are not the same.

1.2 Choosing which formal methods are suitable to deal with dependability in industrial applications

Critical CPS must properly deal with mixed-criticality and heterogeneity (tools, languages, components, teams). The workshops discussion debated reinforcing safety and, more generally, dependability, and which techniques were suitable to being adopted in industrial applications.

* De-CPS organisers

The blanket hypothesis we assumed is a component-based design, largely used in academia and at varying degrees in the industry. Dependability techniques ought to be combined with component-based design.

In this direction, a point first discussed is the achievement of pre-certification of components via modular certification. This later entails the reuse of a component (with its safety analysis and documentation) in another system in the same application domain, or reuse of a component (and its safety analysis) in other application domains. The discussion was based on the results and difficulties encountered by the ambitious FP7 OPENCOSS project. The main difficulty concerns the inconsistency and imprecision in the use of natural language in assurance arguments. To overcome it, the suggested strategy is based on the separation between semantics and syntax features.

The second point entailed an open and rigorous debate in the workshop roundtable: Do Petri Nets still provide a suitable formal method to deal with safety analysis?

The state explosion during the analysis is one factor restraining use of this technique in industrial domains. The other discussion points considered how many industries were known to adopt Petri Nets and what have been their industrial applications.

The large part of the attendees do not adopt, or do not appear to have adopted Petri Nets techniques in industrial projects. Although we have witnessed an increase in the hardware performance, the state explosion during the analysis remains too significant to realistically think that we will be able to one day fix it. This consideration suggests to us that one should adopt probabilistic analysis. Some EU projects seem to confirm the use of probabilistic analysis. For example, the PROXIMA FP7 project addresses timing analysis (included worst-case execution time) by adopting probabilistic analysis techniques for many-core to massive multi-core critical real-time embedded systems.

Finally, we briefly discussed the contract-based approach and technique as a means to deal with dependability in critical CPS. This approach is a suitable means to deal with heterogeneity; it raises from several USA and European research projects. The first industrial research validation is starting to be available and is promising.

1.3 Reconciling differences in the semantic interpretations of the same diagram

The question *how to maintain a high level of control of the development process of the system, according to the expected real-time and performance properties* has been heatedly debated.

About ten years ago, the OMG (Object Management Group), an international community that sets standards, launched MARTE (Modeling and Analysis of Real-Time and Embedded Systems) which is compatible with a SysML/UML design of a system. MARTE is a set of stereotypes that allows engineers to specify time constraints and visualize them via diagrams. In many cases, the analyzer tool, which verifies that the system specification meets the time constraints, is implemented by the open-source and free tools MAST and TimeSquare.

The huge complexity of the standard forced the industrial communities to tailor the original version with non-functional properties, for example, GRM, Generic Component Model; HRM, Hardware Resource Modeling; and VSL, Value Specification Language.

The interpretation of diagrams and model elements is an important issue in the use of MARTE for industrial projects. One astonishing piece of feedback is that engineers haven't the same interpretation of a diagram and model elements especially for real-time features. In many cases, it depends on the background of the engineer. A deep knowledge of the UML standard and the confidence with logic are the most diffuse discriminants that lead to a different interpretation. So, the best lesson learnt from the usage of MARTE in an industrial environment was to specify a clear semantic of each MARTE concept used in practice. Based on these shared and clear semantics, it was easier to interpret the design model done by real-time engineers and to analyze them in available analysis tools.

As described in Section 1.2, the use of natural language for certification is also subject to different interpretations in different application domains - or even the same application domain. The difference in the semantics interpretation was then identified as a common problem.

1.4 Facing the heterogeneity of languages, tools, teams and knowledge

Most of the attendees agreed with the theses provided by Antoine B. Rauzy that facing the diverging nature of engineering solutions could be identified as challenges in the coming years.

Analysis of industrial practice tell us that different teams are involved in the design and in the development of a system's architecture, and that happens 'at the same time'. In many cases, the synchronization between teams and systems' architecture occurs via version-based techniques. This practice ought to - as A. Rauzy said - "accept to live with many models, written in different formalisms, assessed in different ways, with different tools". Obviously, although critical CPS naturally entail heterogeneous environments (languages, tool, studied properties), "it does not mean - continued A. Rauzy - that they have nothing in common".

The conclusion is then: to improve the effort in the way forward to a correct combination and 'coordination', to investigate in the common mechanisms and common notions underlying this heterogeneity.

Acknowledgments

We thank all participants of the workshop that have fruitfully contributed to the discussion.

Follow-up Workshop

The success of the workshop and the industrial feedback suggest consideration for a follow-up workshop for 2015.

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