

# ADA USER JOURNAL

Volume 35

Number 2

June 2014

---

## Contents

	<i>Page</i>
Editorial Policy for <i>Ada User Journal</i>	70
Editorial	71
Quarterly News Digest	72
Conference Calendar	94
Forthcoming Events	100
Press Release	
<i>“Ada 2012 Language Standard Published in Springer's LNCS and as Free eBook”</i>	104
Articles	
M. Mefteh, N. Bouassida and H. Ben-Abdallah <i>“Feature Model Extraction from Documented UML Use Case Diagrams”</i>	107
Proceedings of the “Workshop on Mixed Criticality for Industrial Systems” of Ada-Europe 2014	117
R. Davis et al. <i>“PROXIMA: A Probabilistic Approach to the Timing Behaviour of Mixed-Criticality Systems”</i>	118
A. Alonso and E. Salazar <i>“Toolset for Mixed-Criticality Partitioned Systems: Partitioning Algorithm and Extensibility Support”</i>	123
P. Lindgren, D. Pereira, J. Eriksson, M. Linder and L.M. Pinho <i>“RTFM-lang Static Semantics for Systems with Mixed Criticality”</i>	128
M. Jan, L. Zaourar, V. Legout and L. Pautet <i>“Handling Criticality Mode Change in Time-Triggered Systems through Linear Programming”</i>	133
O. Cros, F. Fauberteau, L. George and X. Li <i>“Mixed Criticality over Switched Ethernet Networks”</i>	138
A. Cohen, V. Perrelle, D. Potop-Butucaru, E. Soubiran and Z. Zhang <i>“Mixed Criticality in Railway Systems: A Case Study on Signaling Application”</i>	144
Ada-Europe Associate Members (National Ada Organizations)	148
Ada-Europe Sponsors	Inside Back Cover

# Editorial Policy for Ada User Journal

## Publication

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

## Aims

*Ada User Journal* aims to inform readers of developments in the Ada programming language and its use, general Ada-related software engineering issues and Ada-related activities. 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](http://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](http://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 editorial is being written at the location of the 19th International Conference on Reliable Software Technologies – Ada-Europe 2014, in the heart of Paris, France, after a really intense week. As usual, the five days of the conference provided both very rich technical contents (some of which we start publishing in this issue) as well as a very pleasant networking atmosphere; I would like to congratulate the conference committees for the excellent week we enjoyed!

I also take the opportunity to invite our readers to the next Ada-Europe conference, which will take place in Madrid, Spain, 22-26 June 2015, organised by the Universidad Politécnica de Madrid. More details about Ada-Europe 2015 can be found in the forthcoming events section of the Journal. Also in this section the reader will find updated information on the SIGAda HILT 2014 conference, which will be co-located with the SIGPLAN SPLASH conference, in Portland, Oregon, October 18-21, 2014. And I am very pleased to inform about the next edition of the International Real-Time Ada Workshop, which will take place April 2015, in Vermont, USA. We expect to publish the call for papers of IRTAW 2015 in the next issue of the Journal.

Concerning the technical contents of this issue, the first paper, from a group of authors from the Sfax University, Tunisia and King Abdulaziz University, Saudi Arabia, presents an automated approach to extract the features model from software product lines.

Afterwards, the issue publishes the Proceedings of the “Workshop on Mixed Criticality for Industrial Systems”, which took place at Ada-Europe 2014, June 27. The workshop provided a discussion forum on research and practice in Mixed-Criticality Systems, showing how research activities interact with industrial needs. The workshop program consisted of a keynote talk, and two technical sessions.

The keynote talk was given by Albert Cohen, senior research scientist in the PARKAS group at INRIA, France on the topic of “Correct-by-Construction Multiprocessor Programming: A Common Approach for Parallel and Mixed-Critical System Design”. Multicore and mixed-criticality are important topics, being even more important to provide safe ways to develop such kind of systems.

The first technical session started with a position paper on the PROXIMA project, on the topic of probabilistic approaches for timing behaviour of mixed-criticality systems, from a group of authors from multiple institutions: the University of York, UK, University of Padova, Italy, Aeroflex Gaisler, Sweden, Sysgo, France, Rapita Systems, UK, Ikerlan, Spain, Airbus Operations, France, INRIA Paris-Rocquencourt, France, Astrium, France, Infineon Technologies, UK and Barcelona Supercomputing Center, Spain. Then, a work from Universidad Politécnica de Madrid, Spain, provided insights in the support for automatic partitioning generation in the MultiPARTES toolset. The third work, from the Luleå University of Technology, Sweden and the CISTER Research Centre, Portugal, described the RTFM-lang approach using compile-time analysis to distinguish critical and non-critical functions and generate the appropriate access mechanisms.

The second session of the workshop started with a work from CEA-LIST, France, Virginia Tech, USA and Telecom Paristech, France, showing how a linear programming approach can be used to generate time-triggered schedules for mode changes in dual-criticality systems. After that, a work from ECE Paris and University Paris-Est, France, analysed how to integrate criticality handling in the IEEE 1588 Precision Time Protocol. And finally, a work from INRIA, Technological Research Institute SystemX, and Alstom Transport, France, presented a case study of mixed-criticality systems, in a signalling railway application.

It was a very rich workshop, with fruitful interaction between participants promoting collaboration between the different communities represented. A very interesting day at the end of the Ada-Europe week.

*Luis Miguel Pinho*  
*Paris*  
*June 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 Events	72
Ada Semantic Interface Specification (ASIS)	73
Ada-related Resources	74
Ada-related Tools	74
Ada-related Products	81
Ada and Operating Systems	82
References to Publications	84
Ada Inside	84
Ada in Context	85

---

## Ada-related Events

### Ada-Europe 2014 in Paris

*From: Dirk Craeynest*

*<Dirk.Craeynest@cs.kuleuven.be>*

*Date: Sun, 4 May 2014 21:29:23 +0200*

*Subject: 19th Int. Conf. Reliable Software Technologies, Ada-Europe 2014*

*To: Ada-Europe-attendees@cs.kuleuven.be*

-----  
Call for Participation

\*\*\* PROGRAM SUMMARY \*\*\*

19th International Conference on Reliable Software Technologies - Ada-Europe 2014

23-27 June 2014, Paris, France

<http://www.ada-europe.org/conference2014>

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

\*\*\* Online registration open! \*\*\*

All info available on conference web site

Early registration discount until May 31

-----  
The 19th International Conference on Reliable Software Technologies - Ada-Europe 2014 takes place in Paris, France, from June 23 to 27, 2014. It is an exciting event with an outstanding technical program, keynote talks, and exhibition from Tuesday to Thursday, and a rich program of workshops and tutorials on Monday and Friday.

The conference is hosted by ECE, a French engineering school located near

the Tour Eiffel, right in the heart of Paris, with convenient connections to all places of interest, and lots of facilities around.

An event not to be missed!

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

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

Also check the conference web site for registration, accommodation and travel information.

Quick overview

- Mon 23 & Fri 27: tutorials

- Tue 24 - Thu 26: core program

Proceedings

- published by Springer

- volume 8454 in Lecture Notes in Computer Science series (LNCS)

- will be available at conference

Program co-chairs

- Laurent George, LIGM/UPEMLV - ECE Paris, France  
[lgeorge@ieee.org](mailto:lgeorge@ieee.org)

- Tullio Vardanega, University of Padova, Italy  
[tullio.vardanega@unipd.it](mailto:tullio.vardanega@unipd.it)

Invited speakers

- Robert Lainé, "Lessons Learned and Easily Forgotten", drawing from his many years of experience in space projects leadership at the European Space Agency and EADS Astrium.

- Alun Foster, "From ARTEMIS to ECSEL: Growing a Large Eco-System for High-Dependability Systems", about the results achieved in ARTEMIS and the objectives of the new ECSEL program, as Acting Executive Director and Programme Manager of the ARTEMIS JU.

- Mohamed Shawky, "Future Challenges in Design Tools and Frameworks for

Embedded Systems; Application to Intelligent Transportation Systems", presenting his futuristic work at the Université de Technologie Compiègne.

Workshops (full day)

- Workshop on "Challenges and new Approaches for Dependable and Cyber-Physical Systems Engineering" (De-CPS 2014), organized by CEA and Thales

- Workshop on "Mixed Criticality Systems" (WMCIS 2014): Challenges of Mixed Criticality Approaches and Benefits for the Industry, organized by ECE

Workshop (half day)

- "Ada 2012: le point sur le langage" (Ada 2012: Assessing the Language), a special session in French for software managers who want to learn about the current state of Ada, organized by Ada-France.

Tutorials (full day)

- "Robotics Programming", Lars Asplund, Asplund Data, Sweden

- "Introduction to Verification with SPARK 2014", Rod Chapman, Altran UK, Yannick Moy, AdaCore, France

Tutorials (half day)

- "Proving Safety of Parallel/Multi-Threaded Programs", Tucker Taft, AdaCore, USA

- "Multicore Programming using Divide-and-Conquer and Work Stealing", Tucker Taft, AdaCore, USA

- "Debugging Real-time Systems", Ian Broster and Andrew Coombes, Rapita Systems, UK

- "Developing Mixed-Criticality Systems with GNAT/ORK and Xtratum", Alfons Crespo, Universidad Politécnica de Valencia, Alejandro Alonso, Universidad Politécnica de Madrid, Jon Pérez, Ikerlan, Spain

- "High-Integrity Object-Oriented Programming with Ada 2012", Ben Brosgol, AdaCore, USA

- "Ada 2012 (Sub)type and Subprogram Contracts in Practice", Jacob Sparre Andersen, JSA Research & Innovation, Denmark

- "Technical Basis of Model Driven Engineering", William Bail, The MITRE Corporation, USA

- "An Overview of Software Testing with an Emphasis on Statistical Testing", William Bail, The MITRE Corporation, USA

#### Papers and Presentations

- 12 refereed technical papers in sessions on Formal Methods, Uses of Ada, Real-Time Scheduling, Applications
- 6 industrial presentations in sessions on Ada in Aerospace, Ada in Railways
- 3 presentations in special "Experience Report" session
- submissions by authors from 22 countries, and accepted contributions from Austria, Canada, Denmark, France, Germany, Italy, Portugal, Republic of Korea, Spain, Sweden, UK, and USA

#### GNAT Retrospective

- 20th anniversary of GNAT as a supported open-source Ada compiler
- started new era for distribution and promotion of Ada language
- retrospective will look back at these important 20 years

#### Vendor exhibition

- 5 exhibitors already committed: AdaCore, Altran, Ellidiss Software, Rapita Systems, and Squaring Technologies; others expected to confirm soon
- vendor presentation sessions in core program

#### Social events

- each day: coffee breaks in the exhibition space and sit-down lunches offer ample time for interaction and networking
- Tuesday evening: Welcome Party
- Wednesday evening: Cruise and Conference Banquet, the traditional Ada-Europe banquet will be on board an all-glass luxury boat, cruising along the Seine right in the heart of Paris!

#### Registration

- early registration discount up to Saturday May 31, 2014
- additional discount for academia, Ada-Europe, ACM SIGAda, SIGBED and SIGPLAN members
- a limited number of student discounts is available
- registration includes copy of printed proceedings at event
- includes coffee breaks and lunches
- three day conference registration includes all social events
- payment possible by credit card, check, or bank transfer
- see registration page for info on novel student waiver program!

Please make sure you book accommodation as soon as possible.

Paris will be very busy in that week.

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

<<http://www.ada-europe.org/conference2014>>.

## Ada Semantic Interface Specification (ASIS)

### ASIS to XML Tools

*From: Robert A Duff  
<[bobduff@shell01.TheWorld.com](mailto:bobduff@shell01.TheWorld.com)>  
Date: Thu, 20 Mar 2014 11:27:44 -0400  
Subject: Re: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

> [...]

The latest GNAT Pro ASIS fully supports Ada 2012. I don't know if that has made it into the GPL version yet.

It will likely be updated to support Ada 2022 (or whatever it will be) when that comes along.

> translates (using ASIS GPL 2012 and my ASIS2XML) to

I didn't know about your ASIS2XML project until now.

Do you know about gnat2xml? It is a similar tool produced by AdaCore. (I wrote it.) It is based on ASIS, and supports Ada 2012. Looking at:

<http://gnat-asis.sourceforge.net/pmwiki.php/Main/ASIS2XML>

I see some differences:

- gnat2xml has cross-links. E.g. each name points to the declaration it denotes, and each expression points to its type.

- There is an XML schema, automatically generated by an ASIS-based tool called gnat2xsd.

- Each XML element has a "source location", which tells you the starting and ending line and column numbers for the corresponding source text. The root of the tree has various information, including the name of the source file.

- There is also a mode in which gnat2xml generates XML interspersed with Ada source text, including comments.

I think XML is horrible. But it has the advantage of being standard, and everybody uses it, and there are all sorts of useful XML-based tools out there.

*From: Robert A Duff  
<[bobduff@shell01.TheWorld.com](mailto:bobduff@shell01.TheWorld.com)>  
Date: Fri, 21 Mar 2014 12:07:39 -0400  
Subject: Re: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

> [...]

That's the xml2gnat tool, which back-translates XML into Ada. It was originally developed for testing purposes: Ada-->XML--Ada ought to produce an Ada program that has identical output to the original.

Then I rewrote the pretty-printer (gnatpp) to use that same Ada-generating code. gnatpp does not use XML, but almost all of the code in xml2gnat is shared by gnatpp.

You could use xml2gnat on modified XML, but you would have to make sure the XML looks like what gnat2xml would generate from some legal Ada. Validating it against the schema using xmllint would help with that. But nobody at AdaCore has ever done that; we use xml2gnat purely for testing gnat2xml. For example, we run all the executable ACATS tests that way (translate the test into XML, then back into Ada, then compile and run the generated Ada, and the output should be identical to the output of the original ACATS test, with the usual "=====  
PASSED ======" message).

The first version of xml2gnat left out all the comments, which aren't needed for the above kind of testing. But of course gnatpp can't leave out comments, so now xml2gnat also includes the comments, using the same shared code.

[...]

### The Future of ASIS

*From: Jean-Pierre Rosen  
<[rosen@adalog.fr](mailto:rosen@adalog.fr)>  
Date: Thu, 20 Mar 2014 15:17:26 +0100  
Subject: Re: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

> [...]

Since there is currently only one implementation of Ada 2012, and since this implementation's ASIS is following the compiler, the decision was taken that the only sensible path was to let GNAT do the work, and standardize on that.

*From: Jean-Pierre Rosen  
<[rosen@adalog.fr](mailto:rosen@adalog.fr)>  
Date: Fri, 21 Mar 2014 06:23:35 +0100  
Subject: Re: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

> Hopefully though, the ASIS WG will come out with the proper specs soon.

That's the problem: there is no ASIS WG anymore. The ARG tried to take over, but it's resources are limited, and better used to the maintenance of the language.

> And FYI, another reason why we need another competing implementation :D

As far as ASIS is concerned, the Gela project was promising. The idea was to have a compiler targeting ASIS, and then

make code generation from ASIS. It was sufficiently advanced that AdaControl compiled with it (except for GNAT extensions for Ada 2005/2012), and even passed a good part of its test suite.

Unfortunately, the author turned to other occupations. It would be nice to have volunteers taking over this project.

*From: Randy Brukardt*

*<randy@rrsoftware.com>*

*Date: Fri, 21 Mar 2014 17:51:12 -0500*

*Subject: Re: Kickstarter for beginning work on a new open-source Compiler*

*Newsgroups: comp.lang.ada*

> [...] Have any private documents made public so that we or others who want to try to implement it, can do? That is, if there are any that are still private.

No. ASIS (unlike the Ada standard) is owned by ISO, and we can't give away their "property". In any case, the updated ASIS standard draft was effectively rejected by the ASIS implementors (by ignoring it and doing their own thing). Vendors have told us outright that they don't think there is any value to an ASIS standard and that they won't support one if it is made. As such, there is no point in putting any more effort into it.

I suggest considering ASIS past Ada 95 as implementation-specific. Perhaps someday there will be agreement on a way forward, but it really only matters for people that want to port ASIS tools. At this point, that community appears to exist solely of J-P Rosen -- everybody else seems to be using GNAT ASIS only (or one of the Atego ASIS implementations only). So there's little point to a standard.

If you disagree, then you need to form your own ASIS WG, put pressure on ASIS vendors to define a consistent set of extensions and ways to plug the many holes in the ASIS standard, and then it's likely that the energy to actually update the standard will materialize. The key here is the pressure on vendors from customers -- they have to see a benefit from having a standard -- they don't see that now.

*From: Tero Koskinen*

*<tero.koskinen@iki.fi>*

*Date: Tue, 25 Mar 2014 17:16:43 +0200*

*Subject: Re: Kickstarter for beginning work on a new open-source Compiler*

*Newsgroups: comp.lang.ada*

>>>> There is also GELA ASIS:

[http://gela.ada-ru.org/gela\\_asis\\_ug](http://gela.ada-ru.org/gela_asis_ug)

>>> Unfortunately, still incomplete and not making progresses any more.

Yes, the project itself is kind of alive (or zombie at least?). Source repository saw commits just 9 days ago:

<http://forge.ada-ru.org/gela/browser>

And the latest release (0.3.2) is from December 2013:

<http://www.ada-ru.org/files/gela-asis-0.3.2.tar.bz2>

I guess the author is not that active on keeping the documentation up-to-date. :)

(PS: Most of this was provided via #ada@Freenode IRC channel. It is quite good information source for all kinds of informal "news".)

---

## Ada-related Resources

### Repositories of Open Source Software

*From: Jacob Sparre Andersen*

*<jacob@jacob-sparre.dk>*

*Date: Sun May 11 2014*

*Subject: Repositories of Open Source software*

AdaForge: 8 repositories [1]

Bitbucket: 109 repositories [2]

16 developers [2]

Codelabs: 18 repositories [3]

Free(code): 79 [4]

GitHub: 543 repositories [5]

127 developers [6]

Rosetta Code: 596 examples [7]

27 developers [8]

Sourceforge: 229 repositories [9]

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

[2] [http://edb.jacob-sparre.dk/Ada/on\\_bitbucket](http://edb.jacob-sparre.dk/Ada/on_bitbucket)

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

[4] <http://freecode.com/search?page=1&submit=Search&with=2880>

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

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

[7] <http://rosettacode.org/wiki/Category:Ada>

[8] [http://rosettacode.org/wiki/Category:Ada\\_User](http://rosettacode.org/wiki/Category:Ada_User)

[9] <http://sourceforge.net/directory/language%3Aada/>

[See also "Repositories of Open Source Software", AUJ 35-1, p. 6. —sparre]

---

## Ada-related Tools

### Bound-T: RAM Usage Analysis for AVR-Ada

*From: Vinicius Franchini*

*<viniciusnf@gmail.com>*

*Date: Sun, 19 Jan 2014 10:18:18 -0800*

*Subject: Ram Usage*

*Newsgroups: comp.lang.ada*

I'd like to know if there are any way to know how much of ram is been used by my code.

I tried the avr-size, but it gives you just the static ram. The problem is how to evaluate the full memory consumption, including the non-static part.

*From: Niklas Holsti*

*<niklas.holsti@tidorum.fi>*

*Date: Mon, 20 Jan 2014 11:40:15 +0200*

*Subject: Re: Ram Usage*

*Newsgroups: comp.lang.ada*

> [...]

For stack consumption you can try a static analyzer such as my Bound-T ([www.bound-t.com](http://www.bound-t.com)). This analyzer does not (yet) address the GNAT secondary stack, but, if I remember correctly, there is no secondary stack in GNAT for AVR. I have no suggestions for heap memory usage.

[Since January 2014 Bound-T has been Open Source software. —sparre]

### Markdown to HTML

*From: Natasha Kerensikova*

*<lithiumcat@gmail.com>*

*Date: Sun, 19 Jan 2014 22:26:04 +0000*

*Subject: RFC: markdown to HTML library*

*Newsgroups: comp.lang.ada*

I have been working on an Ada library that parses lightweight markup languages and render them in various output format (somewhat like pandoc, except I'm not sure my architecture scales easily to a feature set as big as pandoc's).

I wanted to integrate it in the server for my website and let it run in production for a while before formally realising it, however for various reasons it may take a while before I reach that point.

Currently, the library is fully functional with only Markdown front-end and (X)HTML back-end, it passes the official markdown test suite (that I don't distribute because of licence uncertainty) and a decently-covering homegrown test suite (according to gcov, it covers 1112 lines out of 1217 in official markdown front-end, 657/732 lines in markdown extensions, and 348/398 lines in (X)HTML back-end).

Since recently there has been discussions here about Ada for the web, and there's even a FOSDEM talk about it, so maybe Markdown-to-HTML is of interest too.

I would be greatly interested in hearing any comment or criticism or event bug reports about it.

Features request are welcome too, though I can't tell for now when I will manage to look into them. Currently reStructuredText front-end and fully-configurable ODT back-end are on my radar. I can get into the details of how it works internally, but

I won't bore you with it if it's not necessary.

The code is released under ISC licence and currently available on github at <http://github.com/faelys/markup-ada> and eventually the "official" fossil repository will be on my aforementioned website.

## POSIX File Descriptors as Streams

*From: Jacob Sparre Andersen*  
*<jacob@jacob-sparre.dk>*  
*Date: Thu, 06 Mar 2014 11:51:59 +0100*  
*Subject: POSIX streams*  
*Newsgroups: comp.lang.ada*

I've just written a package wrapping a POSIX file descriptor as an Ada stream for use at AdaHeads:

<http://repositories.jacob-sparre.dk/posix-streams>

Feel free to reuse it as Beer-Ware or GPLv3.

## ZanyBlue

*From: Michael Rohan*  
*<michael@zanyblue.com>*  
*Date: Sun, 9 Mar 2014 22:08:43 -0700*  
*Subject: ANN: ZanyBlue v1.2.0 Beta Available*  
*Newsgroups: comp.lang.ada*

A new release of ZanyBlue is now available: 1.2.0 Beta. This is an Ada library currently targeting localization support for Ada (along the lines of Java properties) with supporting message formatting and built-in localization for about 20 locales.

The properties files are compiled into Ada sources which are then built with your application and used to access application messages at run-time. The run-time locale is used to select localized messages, if they are available.

The changes for this release are:

- Updates for building with GNAT 2013.
- Moved usage of Generic packages to library level (stricted accessibility checks with GNAT2013).
- Dropped definition of "ld run path" definition (created issues on MacOS).
- Updated CLDR data to v24 Release (see [cldr.unicode.org](http://cldr.unicode.org)).
- Allow localization for +/- characters to be multi-character strings.
- Improved errors messages for invalid `zmbcompile` command line arguments.
- Implemented message filtering for all Print routines.
- Added directory tree level initialization files for `zbtst`.
- Bugfixes, e.g., handling OS LANG values with dashes in the encoding.
- Some documentation updates.

- Added a patch to ZanyBlue-ize GNAT GPS 5.2.1 (released with GNAT 2013). The resultant executable is identical to the standard `gps` but with support for pseudo translation (no real attempt is made to supply localized properties files for GPS).

Please see the project page on Source Forge for download links, documentation, etc,

<http://zanyblue.sourceforge.net>

This project is licensed under a simple BSD style license.

## Adaino

*From: Pablo Rego <pvrego@gmail.com>*  
*Date: Tue, 11 Mar 2014 04:26:10 -0700*  
*Subject: Adaino (2.0) released*  
*Newsgroups: comp.lang.ada*

I am pleased to announce that I released my library for coding Arduino-like boards with Ada using `avr-elf-windows`, in github for public access. Named Adaino, I just released v.2.0.

<https://github.com/pvrego/adaino>

## Request: LEGO Mindstorms EV3 Development Kit

*From: Karen Sarkisyan*  
*<karen.sarkisyan@gmail.com>*  
*Date: Tue, 11 Mar 2014 02:02:41 -0700*  
*Subject: Ada for Mindstorms EV3*  
*Newsgroups: comp.lang.ada*

I'm interested to know if there are some examples of Ada code or developments for the new version of Lego Mindstorms (EV3 Brick). I know that there is much done for NXT, but the question is about present/future of Ada on this new platform. Any ideas? As far as I know, NXT development resources are incompatible with EV3.

Something from AdaCore to use (Open Source)?

## Emacs Ada Mode

*From: Stephen Leake*  
*<stephen\_leake@stephe-leake.org>*  
*Date: Sat, 15 Mar 2014 08:09:29 -0500*  
*Subject: Emacs Ada mode 5.1.1 and wisi 1.0.2 released*  
*Newsgroups: comp.lang.ada*

Emacs Ada mode 5.1.1 and wisi 1.0.2 are now available in Gnu ELPA, and on my web page:

<http://stephe-leake.org/emacs/ada-mode/emacs-ada-mode.html>

See the NEWS file there for release notes.

The main changes are support for Emacs 24.2 (for Debian stable) and better support for aspects. There are also several bug fixes.

## Ada EL

*From: Stephane Carrez*  
*<Stephane.Carrez@gmail.com>*  
*Date: Wed Mar 19 2014*  
*Subject: Ada EL 1.5.0 is available*  
*URL: http://blog.vacs.fr/vacs/blogs/post.html?post=2014/03/19/Ada-EL-1.5.0-is-available*

Ada EL is a library that implements an expression language similar to JSP and JSF Unified Expression Languages (EL). The expression language is the foundation used by Java Server Faces and Ada Server Faces to make the necessary binding between presentation pages in XML/HTML and the application code written in Java or Ada.

The presentation page uses an UEL expression to retrieve the value provided by some application object (Java or Ada). In the following expression:

```
#{questionInfo.question.rating}
```

the EL runtime will first retrieve the object registered under the name `questionInfo` and look for the `question` and then `rating` data members. The data value is then converted to a string.

The new release is available for download at <http://download.vacs.fr/ada-el/ada-el-1.5.0.tar.gz>

This version brings the following improvements:

- EL parser optimization (20% to 30% speed up).
- Support for the creation of Debian packages.

## Augusta

*From: Peter C. Chapin*  
*<PChapin@vtc.vsc.edu>*  
*Date: Wed, 19 Mar 2014 09:24:36 -0400*  
*Subject: Augusta: An open source Ada 2012 compiler (someday?)*  
*Newsgroups: comp.lang.ada*

In another thread Shark8 posted a proposal to build an IDE+compiler for Ada 2012. In his post he notes that having a second open source compiler offering (besides GNAT) for Ada would be good for the Ada community.

I agree.

GNAT is a fine product but it would enrich the eco-system if there were alternatives. Accordingly I started a pet project for myself to build an Ada 2012 compiler from the ground up which I'm calling "Augusta." The project is here:

<https://github.com/pchapin/augusta>

I am not as naive as I probably sound. I fully understand that such a project is massive and not likely to actually ever be completed. Fortunately that's not important to me. The project is just a hobby project and its \*real\* purpose is to provide me with a source of entertainment

in my off hours. It can fulfill that role perfectly well even if it never amounts to anything. This situation also frees me to make design choices that interest me without feeling the need to justify them rationally. For example Augusta will be written in Scala and will target LLVM. I choose these technologies because I like them and I'd like to learn more about them, not because I think they are somehow the "best" or most logical choices.

I have been planning to announce Augusta's existence to the community at some point but right now the project is 99% talk and 1% action (at most) and I had thought to wait until the balance was a little different. However, Shark8's announcement of his IDE proposal made this seem like a reasonable time. I support his desire to develop such tools and who knows... perhaps Augusta can play some role in his project someday.

Right now Augusta is little more than a place holder with some documents outlining my vision for the project. I have set a release date for myself of December 31, 2020 in an effort to apply some structure to my work. My hope is to have something "interesting" done by that time... although I'm not going to claim it will be full Ada 2012.

In the meantime I've been using Augusta as a source of class examples and student exercises in a compiler course I'm teaching at Vermont Technical College. The work there has been in a sub-project called Allegra which is intended to be a compiler for a series of highly reduced Ada subsets with increasing complexity. In addition to supporting my course, my thought was to use Allegra as a kind of experimentation space for the technologies that will ultimately be part of Augusta. However, I'm not clear how much, if any, of the methods used in Allegra would actually transfer to the more complex Augusta project itself.

Anyway, enough said... I invite anyone who is interested to browse around in the project. Let me know if you have any questions or comments.

[See also "Augusta", AUJ 34-2, p. 68. —sparre]

*From: Tero Koskinen*

*<tero.koskinen@iki.fi>*

*Date: Wed, 19 Mar 2014 20:56:46 +0200*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

[...] it is nice to always see someone to start a new open source Ada compiler to "compete" with GNAT (or just for hobby), however in the history all of the earlier attempts have sadly failed. The Open Source Ada community may be too small and diverse for such a project.

One alternative approach could be to persuade one of the existing Ada vendors

to open source their compiler; perhaps even create a (kickstarter) project to collect the money required for it. (Like it was done for Blender.[1])

[1] [https://en.wikipedia.org/wiki/Blender\\_%28software%29#History](https://en.wikipedia.org/wiki/Blender_%28software%29#History)

*From: Brian Drummond*

*<brian@shapes.demon.co.uk>*

*Date: Wed, 19 Mar 2014 23:04:50 GMT*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [<https://github.com/pchapin/augusta>]

Good luck!

> [...]

LLVM appears to have problems supporting nested (locally declared) subprograms. This appears to be behind slow progress on the Dragonlace project, to use Gnat as an LLVM front end.

Tristan Gingold has recently added an experimental LLVM interface to GHDL (a VHDL compiler; I strongly believe Ada and VHDL users should talk to each other more than they do!) and he also ran into this.

As VHDL makes heavy use of parallel processes, he indicated he would reuse his implementation of processes - essentially closures - to support local subprograms. I don't know the details of how he does this.

(The other advantage of gcc as a backend is that it opens up many more target processors. It would be nice to be able to support both, and the code required in GHDL to support both is really not very much)

*From: Jean-Pierre Rosen*

*<rosen@adalog.fr>*

*Date: Thu, 20 Mar 2014 11:49:27 +0100*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [...]

All those who have been involved in an Ada compiler will tell you that it is a lot more difficult than it appears, unless you stick to the Pascal subset and don't care for validation.

In the early days of Ada, we have seen compilers announcing proudly that they passed 95% of the validation and that delivery was expected in a few weeks - they never succeeded to pass the remaining 5%.

For example, and as a test, make sure you are able to understand the implications of 4.3.3 (a nightmare for code generation), or 13.14, or 3.10.2(3/2)...

*From: Peter C. Chapin*

*<PChapin@vtc.vsc.edu>*

*Date: Fri, 28 Mar 2014 07:31:36 -0400*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [...]

One reason why I choose Scala as an implementation language for Augusta is because I wanted to see if I could productively take advantage of Scala's functional features when writing a serious compiler. As much as I like Ada, I don't think Ada is the most wonderful compiler implementation language imaginable.

For example Scala's algebraic data types and pattern matching make processing trees quite enjoyable and compilers tend to use a lot of trees. Also Scala has good support for creating what the community calls "internal domain specific languages." See for example Graph for Scala (<http://www.scala-graph.org/>), a library for manipulating graphs (perhaps control flow graphs?) in a arguably elegant way. Finally, of course, there might be interesting ways to use higher order functions. I won't know until I try.

My intention has also been to use LLVM or something similar (C, the JVM, etc) as a back end to reduce the amount of work involved in actually getting executable code generated. I understand it is still necessary to generate code for whatever target I use, but the targets above are all higher level than machine language and so (I anticipate) easier to manage.

*From: Randy Brukardt*

*<randy@rrsoftware.com>*

*Date: Mon, 31 Mar 2014 18:43:28 -0500*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [...] As much as I like Ada, I don't think Ada is the most wonderful compiler implementation language imaginable.

But of course Ada is the most wonderful language for everything. :-)

Seriously, though, Ada's strengths are in building large long-lived systems, and any serious attempt to build an alternative Ada compiler has to plan to be a large, long-lived system. Add in the advantages in "eating your own dogfood", and it seems like the obvious choice for such a project.

It seems to me that you have other goals, which is of course fine, but it does a disservice to hold up something that's not really designed for long life as a true alternative system.

> [...]

What's sad (to me) about this is that Ada has added a lot of stuff since Ada 83 that might be very helpful to implementing an structuring a compiler. But all of the existing Ada front-ends originated in Ada 83 and don't use much in the way of Ada 95 features much less newer versions. (It simply doesn't make sense to rewrite large portions of a compiler's code just because one could -- a lot of the time, rewrites don't actually end up any better than the code they replaced, just swapping a set of known problems for unknown ones.)



For instance, Ada 2012 has tree containers that exist in large part because of how common such structures are in compilers and similar applications (like XML and HTML). It would be interesting, for instance, to see if an expression tree written using a class-wide node type stored in an indefinite tree container could be efficient enough to use in a compiler implementation. I'd probably at least consider such a structure (which would eliminate all storage management from being a concern) rather than a access-and-variants that we use. Maybe it wouldn't be better, but it would be different.

It also would be nice to predicates, preconditions, null exclusions, and so on, all of which can allow errors to be detected earlier and easier. (Janus/Ada has many self-checking features, but those all require some work on use by the programmers - it would be better to do that just at writing.)

My point is that there is a lot of the possibilities of Ada for compiler construction (especially of Ada 2012) that hasn't really been explored.

(I'm dubious that pattern matching has much to do with the construction of a compiler front-end, either; that's almost exclusively the province of optimization and code generation, the parts of the compiler you're not planning to work on - even though that's the fun part. :-)

## Starting on a new Open Source Compiler and IDE

*From: Edward R. Fish  
<onewingedshark@gmail.com>*

*Date: Tue, 18 Mar 2014 16:23:57 -0700  
Subject: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

It is my belief that a new, non-GNAT, open-source [and free] Ada 2012 translator would be a good thing for both the Ada community and the general programming population -- this without even breaking from the traditional approaches. However, I think that a good, quality IDE/PSE could be quite advantageous; offering better project-management, documentation, verification, versioning, and correctness/consistency checking.

The working-copy of the proposal is here:

<https://drive.google.com/file/d/0BwQVNNshW39cTXVodWxQaVJ5WjA/edit?usp=sharing>

The link to the Kickstarter is here:

<https://www.kickstarter.com/projects/871049686/squid-open-source-compiler-and-ide-for-the-ada-201>

*From: Maciej Sobczak  
<maciej@msobczak.com>  
Date: Wed, 19 Mar 2014 02:06:05 -0700  
Subject: Re: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

> [...]

Short version: this will fail.

Long version follows.

The proposal itself is fundamentally broken. You have to be aware that the biggest challenge for the Ada community is that of industry awareness. That is, most of the programming community never heard about Ada and those few that did do not see any reason to invest their time in it. The proposal for starting a significant project with public funding should focus on the rationale to explain why such an investment makes sense. But your proposal does not even attempt to do that. In fact, I think that the word "Ada" appears there fewer times than, say, "Delphi". Leaving aside my own relationship with Ada ;-), I see no reason whatsoever why I should give my money to this project.

You can see this problem in the following way: what will happen if you replace the word "Ada" with "Java" or "Scala" in this proposal? It will still be the same proposal and will still make as much (or as little) sense. Which means that this proposal is \*not\* about Ada. It is about your opinion about how IDE should look like.

Which brings us to another important flaw. The Ada (or Java or Scala, if you decide to replace words) programming language was \*defined\* in terms of text. It \*is\* a text-based language. If you think it is a problem, you will not fix it by reinventing the IDE. You have to redefine the programming language to break its natural ties the text format, but if you take the text-based language and try to work with it as if there is no text, you will fail. This idea (should I write IDEa?) is not even new, it was already practiced. I have had a misfortune to work with something like this in the past and it was utter crap that prevented programmers from doing their work efficiently, because the whole concept isolated the programmers from the existing tools. You will never solve all problems in the IDE, at best you will end up with something that is miserable at some selected aspect of programming work and that instead of enabling programmers to spread their wings, just pisses everybody off.

Which basically already started by fixing parts of the solution before exploring the problem - I'm talking about the precise selection of tools that you have proposed at the very beginning: Delphi + .NET + Mono + InterBase? Really?

No, REALLY?

I just fail to imagine Ada enthusiasts (and you need \*a lot\* of enthusiasm to make something as significant) running to contribute to this.

Sorry.

*From: Peter C. Chapin  
<PChapin@vtc.vsc.edu>  
Date: Wed, 19 Mar 2014 09:02:02 -0400  
Subject: Re: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

I think the proposal is interesting and I would love to see it succeed. Maciej Sobczak is concerned about the tool isolation that might arise by storing program information in a database of ASTs (or similar). I can see how that might be a problem. An IDE of the proposed nature would definitely require a robust way to import/export traditional source text. A user of such an IDE could then still use traditional version control systems, CI systems, documentation generators, etc.

I understand that environments such as the one proposed have been attempted before. However, just because they didn't work well in the past doesn't mean this attempt must necessarily fail. This attempt could take lessons from the other systems and, perhaps, avoid or work around the issues that caused problems in the past.

The proposal is very ambitious because it provides both a complex IDE and a compiler for Ada 2012. Either one of those projects would be daunting on its own. To the OP: have you considered ways of having the IDE interact with existing compilers? Ada, in particular, does have an ASIS standard that can guide, to some extent, the design of abstract program representations.

I ask this because I have a pet project of my own to write an Ada 2012 compiler from scratch. I will post more about that in another thread to avoid hijacking this one. However, my vision follows the lead of clang, Microsoft's Roslyn C# compiler, and to a lesser extent the Scala compiler. Modern compilers, I believe, should not just slurp up text and output object code. Instead they should be provided as a collection of well documented libraries that can be directly loaded into other applications. This allows applications, such as IDEs, to exchange information with the compiler using abstract representations while at the same time keeping both projects well isolated. See the clang documentation for what I believe is the right way to do it.

Thus as a potential Ada 2012 compiler writer I ask: what sort of APIs would be useful if one wanted to split off the code analysis and generation from the IDE itself and move it into a separately developed library?

*From: Dmitry A. Kazakov*  
*<mailbox@dmitry-kazakov.de>*  
*Date: Wed, 19 Mar 2014 14:48:18 +0100*  
*Subject: Re: ANN: Kickstarter for beginning*  
*work on a new open-source Compiler*  
*Newsgroups: comp.lang.ada*

> [...]

... designed in a [obscene, dying, vendor-locked, the list can easily be continued] language like Delphi. I doubt Ada community will be eager to contribute in Delphi. I did projects in Turbo, Object Pascal and Delphi, never again!

> [...]

Yes, and it is not the compiler alone to interact with the system. No less important components are:

- source code control system
- project management system
- debugger

Project management is especially important for Ada because Ada projects mainly are cross-platform and/or embedded. That require handling of various targets, cross-compilation, linking, uploading (for embedded targets) etc. For example, mere abstraction of OS-specific dynamic linking as done by GNAT project system is a huge task.

*From: Randy Brukardt*  
*<randy@rrsoftware.com>*  
*Date: Wed, 19 Mar 2014 17:11:32 -0500*  
*Subject: Re: ANN: Kickstarter for beginning*  
*work on a new open-source Compiler*  
*Newsgroups: comp.lang.ada*

> [...]

Beyond that, I'd be very suspicious of any supposedly general purpose programming system that couldn't be created in itself. There is a great value to eating your own dogfood (as the saying goes). It gets rid of gross usability problems in a hurry, and it reduces risk in the project (as there cannot be show-stopping bugs in the development tools -- you just have to fix any problems that occur).

It does require care to avoid the Catch 22 situation where an old bug is preventing work using the old compiler and a new bug is preventing work using a new compiler so that there isn't any obvious way to do any work. (That happened to us once in the mid-1980s - I had to fix the bug in the old compiler with a binary patch in order to cut the Gordian knot and continue. Have been much more careful about regression testing before abandoning old compiler versions since.)

And of course, we all know that Ada is the best language for creating large projects. An Ada development system is surely a large project. There's a reason that virtually all Ada compilers are largely written in Ada. (Some share backends with other systems, but I don't know of any Ada frontends not written in Ada.)

*From: Robert A Duff*  
*<bobduff@shell01.TheWorld.com>*  
*Date: Thu, 20 Mar 2014 10:56:46 -0400*  
*Subject: Re: ANN: Kickstarter for beginning*  
*work on a new open-source Compiler*  
*Newsgroups: comp.lang.ada*

>...I don't know of any Ada frontends not written in Ada.)

The AdaMagic front end is written primarily in C, although I wrote an optimization pass for it in Ada when I was at Intermetrics. The run-time system is written in Ada.

AdaMagic is used by Green Hills, ADI, and others.

If I were writing an Ada compiler from scratch, I would write it in Ada, using GNAT at first, and then bootstrap. Bootstrapping removes any licensing concerns -- you can use whatever license you want for your own work.

Of course, writing an Ada compiler and an IDE is many, many years of work, as has been pointed out.

*From: Randy Brukardt*  
*<randy@rrsoftware.com>*  
*Date: Wed, 19 Mar 2014 16:59:48 -0500*  
*Subject: Re: ANN: Kickstarter for beginning*  
*work on a new open-source Compiler*  
*Newsgroups: comp.lang.ada*

> [...]

It has to provide text import to be considered an Ada 2012 compiler. Ada 2012 requires the compiler to process UTF-8 text in specific formats in order to meet the standard. That's a new requirement for Ada 2012, although as a practical matter any Ada compiler has to be able to process the ACATS and thus needs to have some way to import text.

Unless you're insisting on being an island (and that's not what Ada's about, IMHO), you also need to have export, for no other reason than to allow your code to be used on other Ada compilers when necessary.

*From: Randy Brukardt*  
*<randy@rrsoftware.com>*  
*Date: Wed, 19 Mar 2014 17:03:40 -0500*  
*Subject: Re: ANN: Kickstarter for beginning*  
*work on a new open-source Compiler*  
*Newsgroups: comp.lang.ada*

> I ask this because I have a pet project of my own to write an Ada 2012 compiler from scratch.

Ah, another delusional soul. ;-) Take it from someone who seriously followed that path -- there be dragons. ;-) For one thing, if you're at all successful, you'll be stuck there for the rest of your working life. And to get far enough to be at all successful, you'll have to figure out how to deal with lovely things like resolution, visibility, and tyranny of interfaces. Odds are, you'll never get far enough to work on anything interesting.

*From: Randy Brukardt*  
*<randy@rrsoftware.com>*  
*Date: Thu, 20 Mar 2014 18:04:29 -0500*  
*Subject: Re: ANN: Kickstarter for beginning*  
*work on a new open-source Compiler*  
*Newsgroups: comp.lang.ada*

> [...]

> With the "tyranny of interfaces" do you mean in-general like a generic's formal parameters and a subprogram's signature? Or do you mean the construct of the 'interface' keyword?

The interface construct itself. The issues with actually implementing multi-dispatching are quite daunting, especially given the other requirements of Ada. It clearly can be done; whether I could actually do it is a much more interesting question (particularly without abandoning other properties of our implementation).

*From: Stephen Leake*  
*<stephen\_leake@stephe-leake.org>*  
*Date: Wed, 19 Mar 2014 09:25:39 -0500*  
*Subject: Re: ANN: Kickstarter for beginning*  
*work on a new open-source Compiler*  
*Newsgroups: comp.lang.ada*

[...]

1) Why write the new tool in Delphi and not Ada?

The Rationale section "Why implement in Delphi 2007" provides a rationale for implementing in dotnet, not Delphi. There is an Ada compiler for dotnet (I think it needs work, but you are already proposing a huge amount of work, so what's a little more?).

And even that rationale is not convincing. Emacs, Eclipse, GPS are all portable to Linux and Windows; in what other environments do you require running the IDE?

2) The Rationale section "Why Write an Ada Compiler?" provides a justification for writing an IDE plugin, not an Ada compiler.

You can easily extract text from the database and feed it to gnat, or any other Ada compiler.

3) Storing text as correct, structured data:

- a) has been tried before; in Rational R1000 (<http://en.wikipedia.org/wiki/R1000> - not a very informative link, unfortunately). Did you review any lessons learned from that?
- b) Prevents people checking in code so colleagues can answer the question "why doesn't this compile"?
- c) Prevents writing skeletons; something I do when starting a totally new project.

4) You don't address "Why write a new IDE rather than a plugin for an existing one?"

IDEs are huge, complex beasts. What you propose to do can be accommodated as a plugin for Eclipse, Emacs, or GPS.

GPS from AdaCore is a nice Ada IDE, but it lacks many features that I find essential on a daily basis, which is why I use Emacs instead. Any new tool you start will be even more limited.

5) Current IDEs (GPS, Eclipse, less so Emacs) use a parsed representation of the source for refactoring, completion, and the other tasks you require in an IDE. That gives the best of both worlds.

6) What is your business plan? You are proposing to directly compete with AdaCore; it is not at all clear from the proposal that you understand what that means.

*From: Brian Drummond  
<brian@shapes.demon.co.uk>  
Date: Wed, 19 Mar 2014 22:49:45 GMT  
Subject: Re: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

> One reason to use Delphi rather than Ada is licensing; [...]

Look into that a little more closely ... it is not the case that code \*compiled\* with a GPL compiler is itself GPL'ed.

However it \*is\* the case that code \*linked\* (statically) with a pure GPL runtime system is GPL if the runtime does not have the old "gnat modified GPL" license (aka GMGPL) making an exception by permitting linking to the runtime without extending GPL to the entire executable.

The Libre version of GNAT no longer has GMGPL so you cannot use its runtime in a non-GPL executable.

However, you can build a "zero footprint" Ada program ( without the RTS) as is done for very small embedded MPUs, Atmel AVR, MSP430 etc. and (unless I misread the licensing) these should be GPL-free.

Which means you could in principle substitute another RTS licensed under another license (GMGPL or GPLv3 with the equivalent runtime exception) and link your program (in this case, your own compiler) to that RTS.

And you have to provide such an RTS anyway.

So to use GNAT to bring up your compiler, write the RTS first.

Here's another alternative Ada compiler project...

<https://sourceforge.net/projects/hacadacompiler/>

*From: Luke A. Guest  
<laguest@archeia.com>  
Date: Thu, 20 Mar 2014 03:21:47 -0700  
Subject: Re: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

> One reason to use Delphi rather than Ada is licensing; [...]

There is no reason why you cannot use the FSF GNAT, I use this instead of GPL GNAT for the same reason. Plus I think that GPL is not right for libs.

*From: Stephen Leake  
<stephen\_leake@stephe-leake.org>  
Date: Thu, 20 Mar 2014 18:35:23 -0500  
Subject: Re: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

[...]

> But storing it as unprocessed text means that you [well, your computer/tools] have to repeat a lot of work (e.g. parsing) again and again.

Parsing is very cheap (Emacs Ada mode does it for indentation); the rest of the compiler is hard.

GNAT is open source; you might be able to split out the parsing phase from the rest, and use that parsed representation as the interface between the IDE and the compiler (I have no idea if the GNAT Ada front end is divided that way).

> Non-working code should be kept out of the [main] code-base; this situation can be handled with a chat/message-board (or similar) sort of functionality -  
- There's \*no\* need to pollute your revisions with code that cannot work.

Not true; sometimes the reason it doesn't compile is related to some other change you made. So you need `_all_` of the code. And that's what CM system branches are for.

What is in the main branch of your CM is different than what is in some developer's branch; controlling the flow from developer branches to the main (release) branch is a CM issue, not an IDE source code representation issue.

> You can still have the procedure's "is null" spec; and Delphi itself has been generating empty subprograms forever.

Emacs Ada mode skeletons don't compile in GNAT, but the indentation parser accepts them. For example, 'case foo C-e' expands to:

```
case Foo is
  when =>
end case;
```

I can store that in a file, and since it doesn't compile, I'm reminded to finish it.

Can I store that in your database?

[...]

*From: Simon Wright  
<simon@pushface.org>  
Date: Fri, 21 Mar 2014 08:17:40 +0000  
Subject: Re: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

> case Foo is

> when =>

> end case;

The Rational Environment used to include recognisable and storable markers in such cases: for example {statement}. As far as I remember they were displayed in inverse video to make them stand out!

*From: Erlo  
Date: Mon, 24 Mar 2014 21:31:54 +0100  
Subject: Re: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

>> The problem was (at least for the OS/9 cross) that the programs would compile and link, resulting in an exception (Illegal program, I think) when you tried to execute the code.

> AFAICR the R1000 wouldn't do (the equivalent of) object code generation - "promoting to the code state"? - with placeholders present. But it's a while back!

Been a while for me too! I remember this because it occurred to me that this was a major bummer. It happened more than once that we got these 'illegal program' exceptions on the target caused by placeholders.

When the guys at datamuseum.dk (<http://datamuseum.dk/wiki/Rational/R1000s400>) get their machine running, it's worth a test.

*From: Edward R. Fish  
<onewingedshark@gmail.com>  
Date: Thu, 20 Mar 2014 13:30:44 -0700  
Subject: Re: ANN: Kickstarter for beginning work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

[...]

> 5. This whole project, to me, puts the programmer too far away from the source code. Buries it. That is not a good objective.

That's one problem I'm trying to address, it's not about burying the source, but getting closer to the [underlying] structure and ideas that the code is expressing. An example would be CASE statements, the language mandates that they have complete coverage (individually or via OTHERS) so would it be putting the programmer "too far away from the source" to allow something like "right-click > show as table" to convert the [possibly nested] CASE statement into a decision table? I mean we already have the mandate that all the possibilities must be covered.

Or, imagine having the ability to visually tinker with a [sub]type; say you have Type Voltage is new Natural range 0 .. 220. that you could have represented with two sliders in the same channel (for First and Last) and possibly a tab for indicating [or specifying] the default/uninitialized value. (Perhaps packaging Annex H's "Pragma Normalize\_Scalars" value right there with the type.)

*From: Edward R. Fish  
<onewingedshark@gmail.com>  
Date: Sun, 23 Mar 2014 23:41:54 -0700  
Subject: Re: ANN: Kickstarter for beginning  
work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

[...]

My grand, overarching idea was basically:

- 1) to have the first (and open-source / freely-available) Ada 2012 compiler be written (implementation language rather irrelevant), then
- 2) have the second be written in Ada 2012 (probably w/ SPARK verification/proving) -- hopefully w/ all annexes (though that's a LOT of work) -- and this would be the commercial product, then
- 3) have that compiler/PSE/IDE available for implementing a verified/proved OS.

Like I said upthread; I believe that we /need/ our foundational tools to be without error... and that means investing in verification & correctness checking.

*From: Gautier de Montmollin  
<gautier.de.montmollin@gmail.com>  
Date: Thu, 20 Mar 2014 14:35:02 -0700  
Subject: Re: ANN: Kickstarter for beginning  
work on a new open-source Compiler  
Newsgroups: comp.lang.ada*

Good luck with this ambitious project. A challenge is that at some point the AST knows too much about the program. I've developed a similar system 25 years ago (in Turbo Pascal!) for Ada 83. The pros: no more parsing, compilation is straightforward, links are readily available, the system is able to manage the layout itself, to indent etc. . The cons: formatting rigidity, and if the programmer needs to make a big change, use another package, types and so on, plus all depending changes, he needs to have his program "incorrect" for a while. In my system, it meant export to text, rework the text and then reimport the changed text. Too cumbersome - and tools like GPS offer nowadays navigation tools and autocompletes even in the middle of a such a rework, thanks to smart guessing, and all that at text level...

## Gela

*From: Zhu Qun-Ying  
<zhu.qunying@gmail.com>  
Date: Thu, 20 Mar 2014 11:20:33 -0700  
Subject: Re: Augusta: An open source Ada  
2012 compiler (someday?)  
Newsgroups: comp.lang.ada*

There is another Ada compiler project, and it seems not developed anymore.

<http://gela.ada-ru.org/>

<http://forge.ada-ru.org/gela/wiki>

It is based on the TenDRA compiler.

[See also "The Future of ASIS", earlier in this issue. —sparre]

## ASIS2XML

*From: Simon Wright  
<simon@pushface.org>  
Date: Sun Mar 23 2014  
Subject: ASIS2XML  
URL: <http://asis2xml.sourceforge.net/>*

### ASIS2XML

ASIS is the Ada Semantic Interface Specification; see <http://www.acm.org/sigada/WG/asiswg/>.

This program converts a unit's ASIS representation into XML, so as to make it easier to develop transformational tools using (for example) XSLT.

There is no XML Schema as yet. The output's structure is quite close to that of ASIS, at least in overall terms; for example, an `A_Defining_Name` element in ASIS is represented as a `<defining_name/>` element in XML.

This project was originally hosted on SourceForge as part of ASIS for GNAT, and releases up to 20140413 can be found there.

### Copyright

This work is derived from the `Node_Trav` component of `Display_Source`, which is distributed as a part of the ASIS implementation for GNAT and is Copyright (c) 1995-1999, Free Software Foundation, Inc. The original work in the program is Copyright (c) Simon Wright .

### Licensing

The work is distributed under the terms of the GPL.

### Download

<http://sourceforge.net/projects/asis2xml/files/>

### Prerequisites

- GNAT: GPL 2012 or later, or GCC 4.8 or later
- The corresponding GNAT ASIS
- XML/Ada 1.0 or later

### Building & Use

See the file `INSTALL` in the distribution.

## Excel Writer

*From: Gautier de Montmollin  
<gautier.de.montmollin@gmail.com>  
Date: Sat, 12 Apr 2014  
Subject: Excel Writer v.13  
URL: <http://gautiersblog.blogspot.com/2014/04/excel-writer-v13.html>*

Some recently added features:

- freeze panes
- cell comments
- vertical text alignment
- text orientation
- Ada.Calendar.Time Put/Write and date built-in formats

- background colours
- wrap\_text format option
- Next and Next\_Row
- Text\_IO's New\_Line(lines), Line, Col now available

Excel Writer (Excel\_Out) is a free, standalone, portable, open source package for producing Excel spreadsheets with basic formatings and page layout. It can be used in an "Ada.Text\_IO" fashion, with `Put`, `Put_Line` and `New_Line`.

Download and more informations here: <http://excel-writer.sf.net/>

## GNAT GPL and SPARK GPL

*From: Jamie Ayre <ayre@adacore.com>  
Date: Tue, 13 May 2014 09:46:33 -0400  
Subject: [AdaCore] Announcing the  
immediate availability of GNAT and  
SPARK GPL 2014  
To: [libre-news@lists.adacore.com](mailto:libre-news@lists.adacore.com)*

We are pleased to announce the availability of the GNAT and SPARK GPL 2014 toolsets.

GNAT GPL 2014 incorporates more than 120 new features, including Ada 2012 mode enabled by default, many new warnings and improved diagnostics, code generation optimizations, support for symbolic traceback in shared libraries, and improved cross Ada/C++ exception handling.

GNAT GPL 2014 introduces GNAT2XML, for generating XML files from Ada sources, which will help in writing Ada analysis tools quickly in any language. It also provides enhancements to existing tools, including:

- A new version of GNATpp, providing improved Ada layout and greater flexibility
- Support in the GPRbuild multipurpose builder for distributed builds, and better support for parallel builds

It also comes with the latest version of the GPS IDE. The complete list of the major new features in GPS 6.0.1 is accessible here:

[http://docs.adacore.com/gps-docs/release\\_notes/build/singlehtml/](http://docs.adacore.com/gps-docs/release_notes/build/singlehtml/)

SPARK GPL 2014, supporting SPARK 2014, is the first GPL release of the next generation SPARK toolset.

The main features of the new language and toolset are as follows:

- Convergence with Ada 2012 Syntax
- Bigger Language Subset
- Executable Contracts
- Hybrid Verification (the ability to mix unit proof with unit test)
- Formal Container Library

- Generative Mode for Data Dependencies (the ability to perform data flow analysis without explicit global declarations)
- Improved IDE feedback in relation to information flow and verification errors

You will find documentation about the SPARK 2014 toolset here:

<http://libre.adacore.com/developers/documentation>

See in particular the SPARK 2014 Toolset User's Guide to get started. You can also read more about the SPARK 2014 language - including a growing number of tips and tricks here:

<http://www.spark-2014.org>

Both toolsets can be downloaded from [libre.adacore.com](http://libre.adacore.com).

---

## Ada-related Products

### GNAT Pro

*From: AdaCore Press Center*

*Date: Tue Feb 25 2014*

*Subject: AdaCore Releases GNAT Pro 7.2*

*URL: <http://www.adacore.com/press/gnat-pro-7-2/>*

New major release of Ada software development environment includes 120+ new features and extends support across ARM platforms.

NEW YORK, PARIS, and NUREMBERG, Germany, February 25, 2014 – Embedded World Conference – AdaCore today announced the next major release of its Ada development environment, GNAT Pro 7.2. Embodying the constant innovation that has driven the product's evolution, this latest GNAT Pro toolsuite incorporates more than 120 new features, many of which are based on customer suggestions. This latest GNAT Pro toolsuite includes several new tools, is available on additional platforms, implements the Ada 2012 language standard by default, and extends its coverage of ARM configurations. The wide range of new or improved functionality brings Ada developers the benefits of increased flexibility, greater efficiency, and broader platform support, all within the context of AdaCore's open source technology and unrivaled support.

GNAT Pro is available on more native and cross platforms than any other Ada development environment, and the 7.2 release adds support for Wind River's VxWorks Cert and LynxWorks' LynxOS-178 Real-Time Operating Systems (RTOS). It also extends GNAT Pro's ARM support to now include Android, generic Linux on ARM, Bareboard ARM, and Wind River's VxWorks 6 on ARM.

GNAT Pro 7.2 comes with the GPS (GNAT Programming Studio) 6.0 Integrated Development Environment,

providing developers with more space for editing and a number of design improvements that bring program-related information within easy reach. The revised look and feel is supported by a new relational database at the heart of the GPS engine, making code navigation much more efficient. It also includes a new version of GNATbench, the Eclipse plug-in. GNATbench 2.8 provides improved support for Wind River's WorkBench, a new source navigation engine, and improved support for the CodePeer static analysis tool.

GNAT Pro 7.2 includes several new tools, including GNAT2XML, which generates XML files from Ada sources and helps developers write Ada analysis tools in any language. Enhancements to existing tools include a new version of GNATpp (pretty printer) with improved Ada layout, and an enhanced GPRbuild multi-purpose builder that offers greater flexibility and support of both distributed and parallel builds.

Other new features of GNAT Pro 7.2 include new warnings and improved diagnostics, code generation optimizations, support for symbolic traceback in shared libraries, and improved cross Ada/C++ exception handling.

"With so many new features and tools in GNAT Pro 7.2, it's difficult to choose which to highlight," said Cyrille Comar, AdaCore EU Managing Director. "I'll pick the new, extremely efficient distributed build capability. Its first industrial user reported that the build time for its complete multi-million SLOC application went down from two hours to five minutes on a Linux farm with dozens of machines. This opens the door to a new level of agility in the development of such massive applications!"

### GNAT Pro for ARM/Linux

*From: AdaCore Press Center*

*Date: Tue Feb 25 2014*

*Subject: AdaCore Releases GNAT Pro 7.2 for ARM/Linux*

*URL: <http://www.adacore.com/press/gnat-pro-7-2-arm-linux/>*

New major release of AdaCore's development environment extends support for ARM platforms.

NEW YORK, PARIS and NUREMBERG, Germany, February 25, 2014 – Embedded World Conference – AdaCore today announced the release of its latest Ada cross-development environment, GNAT Pro 7.2, for ARM processors running Linux. This GNAT Pro ARM product provides a complete Ada development environment oriented towards embedded systems that require the flexibility and extensive services provided by Linux. Developers of such systems can now exploit the software engineering benefits of the Ada language,

including reliability, maintainability, and portability.

"Ada and ARM share at least one major characteristic: they provide a combination of strong industrial maturity and innovative adaptability to their ecosystem," said Cyrille Comar, AdaCore EU Managing Director. "While ARM was systematically addressing the whole spectrum of embedded, low-consumption processors, from the smallest microcontroller to the most powerful multicore, Ada was addressing the rising needs of safer programming techniques through its enhanced support of contract programming. Thanks to GNAT Pro's recent and extensive ARM support, it is now possible to benefit from these combined elements simultaneously."

Incorporating more than 120 new features, this latest GNAT Pro toolsuite implements the Ada 2012 language standard by default, and extends its coverage of ARM configurations to complement GNAT Pro products for VxWorks 6 ARM and bare-board ARM. Some of the new Ada 2012 language features include:

- Contract-based programming (preconditions, postconditions, and type invariants).
- In-out parameters for functions (a much-requested enhancement to the language).
- Enhanced multiprocessor support (multiprocessor affinity and barriers).
- Enhanced integration of concurrency and OOP (re-queue on synchronized interfaces).
- Additional language-defined libraries (vector/matrix libraries).

GNAT Pro 7.2 comes with the GPS (GNAT Programming Studio) 6.0 Integrated Development Environment, providing developers with more space for editing and a number of design changes that bring program-related information within easy reach. The revised look and feel is supported by a new relational database at the heart of the GPS engine, making code navigation much more efficient. GNAT Pro 7.2 also includes a new version of GNATbench, the Eclipse plug-in. GNATbench 2.8 provides a new source navigation engine and improved support for the CodePeer static analysis tool.

### SPARK Pro

*From: AdaCore Press Center*

*Date: Tue May 6 2014*

*Subject: Altran and AdaCore Release Next-Generation Static Verification Toolset*

*URL: <http://www.adacore.com/press/next-gen-static-verification-toolset/>*

SPARK Pro 14.0 brings new proof technology and additional language features to developers of high-integrity software.

Altran and AdaCore announce the release of the SPARK Pro 14.0 integrated development and verification environment. This product marks a major step forward in software verification technology, providing users with more powerful and easier to use tools that support the latest version of the SPARK language, SPARK 2014. SPARK Pro 14.0 offers an integrated approach to the entire software development and verification lifecycle – bringing software specification, coding, testing and unit verification by proof within a single integrated framework.

SPARK Pro 14.0 has been completely re-engineered to use the latest compiler and proof technology, providing advanced verification of an enhanced subset of the Ada language. The new technology also supplies an improved user interface: warnings generated by the tools are displayed as navigable messages mapped back to the source code with path information that helps users understand how the errors can occur.

SPARK Pro 14.0 meets the requirements of all high-integrity software safety standards, including DO-178B/C (and the formal methods supplement DO-333), CENELEC 50128, IEC 61508, and DEFSTAN 00-56. The SPARK Pro toolset generates evidence that can be used to build a constructive assurance case and demonstrate conformance to the appropriate standard. SPARK Pro can also be used to help achieve the highest Evaluation Assurance Levels (EAL) of the Common Criteria security standard. Building software that is right the first time avoids the costs associated with extensive test and debug cycles and expensive product failures and recalls.

“Given the widespread use of Intelligent Systems across many sectors, the adoption of the new SPARK 2014 technology makes complete business sense” said Keith Williams, Group Vice President, Intelligent Systems / Altran. “Our clients need to ensure user requirements are met and costly events such as recalls are avoided ... SPARK Pro 14.0 does both”.

SPARK Pro 14.0 is the first version of the toolset to support SPARK 2014, the newest version of the language. SPARK 2014 is based on Ada 2012 and encompasses a rich subset of the language, excluding only those features which would make program verification unsound. SPARK uses and extends the contract notation introduced in Ada 2012, allowing software engineers to express and formally verify key properties that must be satisfied by a program.

“After decades as a niche technology, we have finally reached the stage where formal proof techniques can play an important part in the development of a wide range of software” said Robert

Dewar, co-founder and President of AdaCore. “SPARK Pro 14.0 embodies the new promise of this technology.”

## Ada and Operating Systems

### OpenBSD: Compiler Availability

*From: Tero Koskinen*  
*<tero.koskinen@iki.fi>*  
*Date: Fri, 21 Feb 2014 07:59:57 +0200*  
*Subject: Re: Best book to learn ada?*  
*assuming openbsd 5.4 amd64 box here*  
*Newsgroups: comp.lang.ada*

> [...]

OpenBSD has gnat. Just type

```
pkg_add gnat-4.8.1p1
```

or

```
pkg_add gnat-4.6.4p1
```

and you will have mostly working gnat in your system.

### Windows: Opening a Web Page

*From: Tero Koskinen*  
*<tero.koskinen@iki.fi>*  
*Date: Mon Mar 3 2014*  
*Subject: Opening a web page using default browser with Ada on Windows*  
*URL: http://ada.tips/opening-a-web-page-using-default-browser-with-ada-on-windows.html*

Here is how you can open a web page with the default browser on Windows:

```
with Interfaces.C; use Interfaces.C;
with Interfaces.C.Strings;
use Interfaces.C.Strings;
```

#### procedure Browser is

```
Cmd : aliased char_array := To_C("open");
HTML_Path : aliased char_array :=
    To_C ("http://ada.tips/");
Dir : aliased char_array := To_C ("");
H : Long;
```

```
function Shell_Execute
(Wnd : Int;
 Operation, File, Parameters,
 Directory : chars_ptr;
 ShowCmd : Int) return Long;
```

```
pragma Import (Stdcall, Shell_Execute,
 "ShellExecuteA");
```

#### begin

```
-- ShellExecute(null, "open", htmlpath,
-- NULL, "", SW_SHOWNORMAL);
H := Shell_Execute
(0,
 To_Chars_Ptr (Cmd'Unchecked_Access),
 To_Chars_Ptr
 (HTML_Path'Unchecked_Access),
 Null_Ptr,
 To_Chars_Ptr (Dir'Unchecked_Access),
```

1);  
**end** Browser;

As you can see, you need only need to import Shell\_Execute and call it with "open" parameter and the url you want to see.

### FreeBSD: State of Ada Ports

*From: John Marino*  
*<dragonlace.cla@marino.st>*  
*Date: Sat, 15 Mar 2014 10:55:00 -0700*  
*Subject: State of FreeBSD Ports: All built with GCC 4.9.0 (2014-03-02)*  
*Newsgroups: comp.lang.ada*

State of FreeBSD Ports: All built with GCC 4.9.0 (prerelease, 2014-03-02)

It's been a while since I posted on CLA. I've primarily been working with \*BSD support although I also came up with the first Android cross compiler. As part of the BSD support, I updated FreeBSD Ports to all build with GCC 4.9.0 prerelease (02 March snapshot). In the last few months I've steadily been adding Ada support to FreeBSD Ports (which supports FreeBSD and DragonFly BSD) and I believe that these two platforms are now very strong candidates for Ada development.

As can be seen on resurgent dragonlace.net website, the new FSF GNAT compiler still passes all tests on ACATS and GNAT.DejaGNU suites. I've also been updating available software lists at [en.wikibooks.org/wiki/Ada\\_Programming/Installing#FreeBSD\\_and\\_DragonFly](http://en.wikibooks.org/wiki/Ada_Programming/Installing#FreeBSD_and_DragonFly)

The current versions of Ada ports are as follows:

- archivers/zip-ada (46)  
Zip-Ada (Library)
- devel/adabooc (2013-03-22)  
Ada95 Booch Components (Library)
- devel/adacurses (2011-04-04)  
AdaCurses (Binding)
- devel/afay (41111)  
AFlex and AYACC parser generators
- devel/ahven (2.4)  
Ahven (Unit Test Library)
- devel/florist-gpl (2013)  
Florist (Posix Binding)
- devel/gnatpython (2014-02-05)  
GNATPython (python-based test framework)
- devel/gprbuild (2013)  
GPRbuild (Multi-language build tool)
- devel/gps (5.2.1)  
GNAT Programming Studio
- devel/matreshka (0.6.0)  
Matreshka (Info Systems Library)
- devel/libspark2012 (2012)  
SPARK 2012 library source files-
- devel/sdl\_gnat (2013)  
GNAT SDL bindings (Thin)

- dns/ironsides (2014-02-20)  
Spark/Ada Ironsides DNS Server
- lang/adacontrol (1.15r5)  
AdaControl (Construct detection tool)
- lang/asis (2013)  
Ada Semantic Interface Specification
- lang/gcc-aux (4.9.0-PR)  
GNAT Ada compiler (FSF GCC)
- lang/gcc47-aux (4.7.3)  
GNAT Ada compiler (FSF GCC)
- lang/gnatdroid-armv5 (4.7.3)  
Android 2.3 cross-compiler, ARMv5
- lang/gnatdroid-armv7 (4.7.3)  
Android 2.3 cross-compiler, ARMv7
- math/plplot-ada (5.10.0)  
PLplot Ada bindings
- net/anet (0.2.3)  
Network library (IPv4 and IPv6)
- net/polyorb (2.10.0/2013)  
PolyORB (CORBA/SOAP/DSA  
middleware)
- net/adasockets (1.8.11)  
IPv4 socket library
- security/libsparkcrypto (0.1.1)  
LibSparkCrypto (Cryptography Library)
- textproc/adabrowse (4.0.3)  
AdaBrowse (Ada95 HTML doc.  
generator)
- textproc/opentoken (5.0a)  
Ada Lex analyzer and parser
- textproc/py-sphinxcontrib-adadomain  
(0.1)  
Sphinx Ada docs generator
- textproc/words (1.97F)  
Words (Latin/English dictionary)
- textproc/xmlada (4.4.0)  
XML/Ada (Library)
- www/aws (3.1.0.0w)  
Ada Web Server
- www/aws-demos (3.1.0.0w)  
Ada Web Server demos
- x11-toolkits/gtkada (2.24.4)  
GTK/Ada (bindings)
- x11-toolkits/qtada (3.2.0.0)  
Qt/Ada (bindings)

I'm still improving this list though. GNATDroid should get upgraded to GNAT 4.9.0 soon, and when SPARK 2014 is available I'll look to add that as well. There are also a list of smaller ports that will trickle in.

As one can see, most of these ports are the latest available so FreeBSD and DragonFly BSD deserve serious consideration when looking for a platform that facilitates Ada development.

*From: John Marino*  
*<dragonlace.cla@marino.st>*  
*Date: Sun, 23 Mar 2014*  
*Subject: Four new ports added*  
*URL: [http://www.dragonlace.net/posts/Four\\_new\\_ports\\_added/](http://www.dragonlace.net/posts/Four_new_ports_added/)*

Four new packages have been added to FreeBSD ports collection:

- textproc/xml\_ez\_out
- graphics/generic\_image\_decoder
- misc/ini\_files\_manager
- misc/excel-writer

Three of those are the works of Gautier de Montmollin, and they have been converted into static libraries with dedicated gpr files in the standard GNAT location.

*From: John Marino*  
*<dragonlace.cla@marino.st>*  
*Date: Sat, 12 Apr 2014*  
*Subject: New ports and gnatdroid update*  
*URL: [http://www.dragonlace.net/posts/New\\_ports\\_and\\_gnatdroid\\_update/](http://www.dragonlace.net/posts/New_ports_and_gnatdroid_update/)*

Recently added to the FreeBSD ports collection was codelabs.ch's pscs-ada (thick Ada binding to PC/SC-middleware) and the APQ Ada95 database binding with drivers for MySQL, PostgreSQL, and ODBC included as separate ports.

A huge effort went into updating the GNATDroid ARM cross-compiler to be based on GCC 4.9. This is the only ARM compiler that supports sockets to my knowledge -- socket support is disabled on a stock gcc, but I've got it working and it passes the related testsuite.

The only thing that doesn't pass is the stack-check tests. That is because stack-checking as not yet been implemented for the ARM target on GCC. A patch to add the capability was created but never added, but hopefully it gets added soon.

Other internal improvements include getting the ACATS test to run on a remote device in 15 minutes rather than 6 hours, and to get the gnat.dg testsuite to run for the first time on a remote device. The results are publish on the main page (they look good!)

## Windows: System Tray and Taskbar Manipulation

*From: Gautier de Montmollin*  
*<gautier.de.montmollin@gmail.com>*  
*Date: Thu Mar 20 2014*  
*Subject: Tiny but useful gadgets*  
*URL: <http://gautiersblog.blogspot.com/2014/03/tiny-but-useful-gadgets.html>*

New GWindows packages, for accessing the system tray and the taskbar in a comfortable way from your Ada applications.

- GWindows.System\_Tray
- GWindows.Taskbar

Latest additions to GWindows are available from the SVN repository:  
<http://sourceforge.net/p/gnavi/code/HEAD/tree/>

## Debian: Switching to GNAT 4.9

*From: Ludovic Brenta*  
*<ludovic@ludovic-brenta.org>*  
*Date: Thu, 24 Apr 2014 23:12:29 +0200*  
*Subject: gnat-4.9 4.9.0-1 uploaded to unstable*  
*To: [debian-ada@lists.debian.org](mailto:debian-ada@lists.debian.org)*

Hello, world.

GCC 4.9.0 has been officially released on Tuesday and uploaded to unstable on Wednesday. I just uploaded gnat-4.9 4.9.0-1 to match.

We're three weeks late on the schedule[1] I proposed after FOSDEM but things are looking very good indeed. Several packages, including ASIS and GtkAda, are almost ready for upload. I wish to specially commend the excellent work by Nicolas Boulenguez on several packages that made this possible. In addition, a new and very recent version of polyorb has already been uploaded to the NEW queue, thanks to Xavier Grave.

Therefore, real soon now(tm) I will upload a new version of "gnat" that switches the default Ada compiler to gnat-4.9, which will begin the transition of all packages to this new compiler. Note that this will cause all current Ada packages to be removed from testing. Package maintainers, please start working on this transition as soon as possible if you have not already started.

[1] <https://lists.debian.org/debian-ada/2014/02/msg00000.html>

## Mac OS X: GNAT 4.9.0

*From: Simon Wright*  
*<simon@pushface.org>*  
*Date: Wed, 30 Apr 2014 10:12:08 +0100*  
*Subject: ANN: GCC 4.9.0 for Mac OS X Mavericks*  
*Newsgroups: comp.lang.ada*

GCC 4.9.0, with the GNAT GPL 2013 tools, is available at

[https://sourceforge.net/projects/gnuada/files/GNAT\\_GCC%20Mac%20OS%20X/4.9.0/](https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/4.9.0/)

There will be another release when GNAT GPL 2014 appears.

This is the README:

This is GCC 4.9.0 built for Mac OS X Mavericks (10.9.2, Darwin 13.1.0), with Xcode 5.1.1.

gcc-4.9.0-x86\_64-apple-darwin13-01.tar.bz2

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

Tools included: ASIS, AUnit, GPRbuild, XMLAda from GNAT GPL 2013 and GNATColl from the public Subversion repository.



```
Target: x86_64-apple-darwin13
Configured with: ../gcc-4.9.0/configure \
--prefix=/opt/gcc-4.9.0 \
--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
Thread model: posix
gcc version 4.9.0 (GCC)
```

```
MD5 (gcc-4.9.0-x86_64-apple-darwin13-
01.tar.bz2) = 74229b5339324cd7ef7bbaa
b0316c4bb
```

### Install by

```
$ cd /
$ sudo tar jxvf ~/Downloads/gcc-4.9.0-
x86_64-apple-darwin13-01.tar.bz2
and put /opt/gcc-4.9.0/bin first on your
PATH.
```

### Notes

The compiler is GPL version 3 with the Runtime Exception, so executables built with it can be released on proprietary terms PROVIDED THAT they make no use of the packages from GNAT GPL 2013, which are full GPL.

The command 'gnat', as originally built, failed with SIGSEGV. It was rebuilt on its own, using the project file gnatcmd.gpr, and no longer failed; the working version is provided.

Changes made to GPRbuild GPL 2013 are in gprbuild-2013-src.diff. They:

- remove the '-c' flag that is wrongly passed to ranlib (and isn't by gnatmake).
- correct a problem when building static stand-alone libraries.

GNATColl GPL 2013 wouldn't build. Instead, GNATColl (SVN revision 226851) was configured as below, which is minimal apart from GNU Readline being enabled. Users may wish to reconfigure for their own requirements.

```
./configure \
--prefix=/opt/gcc-4.9.0 \
--build=x86_64-apple-darwin13 \
--enable-gpl
```

resulting in

Shared libraries: yes (default: static)

Gtk+: no (requires pkg-config and gtkada.gpr)

Python: yes

```
/System/Library/Frameworks/
Python.framework/Versions/2.7
(see --with-python)
```

PyGtk: no (see --enable-pygtk)

PyGObject: no (see --enable-pyobject)

Syslog: yes (see --enable-syslog)

Readline (GPL license): yes  
(see --with-readline --enable-gpl)

gmp: no (see --with-gmp)

PostgreSQL: no -L/usr/lib  
(see --with-postgresql)

Sqlite: embedded (see --with-sqlite)

Iconv: yes (see --with-iconv)

Projects: yes

Changes to ASIS GPL 2013 are in asis-gpl-2013-src-4.9.0.diff. Only changes necessary for the build are included.

In addition to the above, a new library gnat\_util is required by ASIS and GNATColl. A Sourceforge project to provide this has been set up at <https://sourceforge.net/projects/gnatutil/>; release 4.9.0 is included here. This is the equivalent of the Debian libgnatvsn.

## References to Publications

### Books for Learning Ada

From: John W. McCormick

<mccormick@cs.uni.edu>

Date: Thu, 20 Feb 2014 12:23:41 -0800

Subject: Re: Best book to learn ada?

assuming openbsd 5.4 amd64 box here  
Newsgroups: comp.lang.ada

I cannot recommend my book "Ada Plus Data Structures" for learning Ada. It is targeted at CS2 (2nd course in computer science) students. It assumes that you already know the control structures and some of Ada's type model. It then teaches the basic data structures. Only a brief introduction to Ada's OO capabilities. Really a beginning level book best paired with my introductory book "Programming and Problem Solving with Ada".

Since I have damned one of my books, I will recommend another that includes an introduction to Ada for folks who have skills in other programming languages.

Building Parallel, Embedded, and Real-Time Applications with Ada

McCormick, Singhoff, and Hugues

Cambridge Press, 2011

Check <http://www.embedded.com/electronics-blogs/break-points/4411676/A-new-Embedded-Ada-book> for a review by Jack Ganssle.

From: Richard Riehle <rriehle@itu.edu>

Date: Thu, 20 Feb 2014 12:59:17 -0800

Subject: Re: Best book to learn ada?

assuming openbsd 5.4 amd64 box here  
Newsgroups: comp.lang.ada

> [...]

Reminder. My book, Ada Distilled, is available on-line. It is designed to help

someone who already can program in another language learn how to program in Ada.

All the examples are fully coded. They will compile. They will execute. It is not completely ready for Ada 2012, but Ed Colbert is working on updating it for the new 2012 standard.

You can download Ada Distilled free from several sites.

From: Jerry Bauck

<lanceboyle@qwest.net>

Date: Fri, 21 Feb 2014 15:22:24 -0800

Subject: Re: Best book to learn ada?

assuming openbsd 5.4 amd64 box here  
Newsgroups: comp.lang.ada

> [...]

My standard reply to this question is Norman H. Cohen's Ada as a Second Language. I have the second edition which is for the 95 standard. I wish it would be updated for newer Ada's but that's not happening. I find his exposition to be outstanding, and he doesn't weigh the book down with lengthy examples or a running book-length example, only short, to-the-point examples.

Amazon lists it new for \$595.15 to \$3,295.38 (U.S.) or used at \$84.85. I'd say used would be your best buy. 8^).

## Ada Inside

### Airbus Helicopters Selects Vector Software as Software Testing Solution Provider

From: Vector Software Press Releases

Date: Tue Apr 22 2014

Subject: Airbus Helicopters Selects Vector Software as Software Testing Solution Provider

URL: <https://www.vectorcast.com/news/vector-software-press-releases/2014/airbus-helicopters-selects-vector-software-software-testing>

VectorCAST Continuous Build and Integration Capabilities Enable Robust Ada and C++ Testing

Vector Software, the world's leading provider of innovative software solutions for testing safety and mission critical embedded applications, announced today that Airbus Helicopters selected the VectorCAST™ solution to test software for the UH-Tiger military helicopter project. The company chose the VectorCAST/Cover, VectorCAST/C++ and VectorCAST/Ada tools to ensure that they are able to use a fully automated regression test environment to continually verify the correctness of their code.

Airbus Helicopters in Germany develops software for the UH-Tiger military helicopter, and the firm needed to quickly achieve DO-178B structural coverage while using limited resources to develop



User Control Panels. The RTCA DO-178B standard is one of the most stringent safety critical standards in the world, incorporating the most rigorous testing and traceability requirements of any industry. In order to meet these goals, Airbus Helicopters selected the VectorCAST/C++ and VectorCAST/Ada tools to obtain the most automated solution available for unit and integration testing of complex C/C++, and Ada code applications.

The UH-Tiger helicopter is a medium weight, multiple role support helicopter developed for the German Armed Forces. The aircraft is known for being the first all-composite, European-built helicopter and includes advanced features like a glass cockpit, stealth technology and tremendous agility for enhanced combat survivability. Airbus Helicopters is using VectorCAST products on projects such as the User Control Panels, and selected the testing solution because of the tool's ease of test case development and execution in addition to its flexible reporting options.

"The VectorCAST tools help Airbus Helicopters ensure that complex applications meet DO178B standards", said Bill McCaffrey, Chief Operating Officer, Vector Software. "Organizations that use automated testing solutions can more easily meet rigorous standards like DO-178B on time, and on budget."

## Ada in Context

### Where to Override Stream Attributes

*From: HP <hanslad@gmail.com>  
Date: Thu, 23 Jan 2014 10:53:23 -0800  
Subject: Binary and XML serialization of types  
Newsgroups: comp.lang.ada*

I am an Ada beginner who is working on a private project. The project is to implement a protocol which either does binary or XML serialization of the different defined records.

I have tried to separate all the "encoding" details from the type declaration in a sub package like this:

```
package A.Types is
  type Guid_Array is array (1 .. 8) of
    Unsigned_8;

  type Guid is record
    Data1 : Unsigned_32;
    Data2 : Unsigned_16;
    Data3 : Unsigned_16;
    Data4 : Guid_Array;
  end record;
end A.Types;

with Ada.Streams; use Ada.Streams;
package A.Types.BinaryEncoder is
  procedure Guid_Write
    (Stream : access Ada.Streams.
     Root_Stream_Type'Class;
     Item   : in Guid) is
begin
  A.Types.BinaryEncoder.Guid_Write
    (Stream, Item);
end Guid_Write;
```

```
Root_Stream_Type'Class;
Item   : in Guid);
for Guid_Write use Guid_Write;
end A.Types.Encoders;
```

I get the following error:

7:8 entity must be declared in this Scope

How can I separate all the encoding and decoding details from the type declaration? I like the idea of splitting this into packages with different functionality. Is this possible at all?

*From: Adam Beneschan  
<adam@irvine.com>  
Date: Thu, 23 Jan 2014 11:15:56 -0800  
Subject: Re: Binary and XML serialization of types  
Newsgroups: comp.lang.ada*

> [...]

I think you simply want to do something like this. Put the declaration of Guid\_Write and the "for" clause the specification of in A.Types. Then, in the body of A.Types:

```
with A.Types.BinaryEncoder;
package body A.Types is
  -- other stuff as needed
  procedure Guid_Write
    (Stream : access Ada.Streams.
     Root_Stream_Type'Class;
     Item   : in Guid)
  renames A.Types.BinaryEncoder.
    Guid_Write;
  -- this is called a "renaming-as-body"
```

or this, which amounts to the same thing:

```
with A.Types.BinaryEncoder;
package body A.Types is
  -- other stuff as needed
  procedure Guid_Write
    (Stream : access Ada.Streams.
     Root_Stream_Type'Class;
     Item   : in Guid) is
begin
  A.Types.BinaryEncoder.Guid_Write
    (Stream, Item);
end Guid_Write;
```

(Note: I think the "renames" will work, but I haven't tested it. The second one will definitely work.)

Now you declare and implement Guid\_Write in A.Types.BinaryEncoder as you were trying to do. (You don't actually need to give it the same name. You can call your "implementation" procedure Guid\_Write\_Impl, or Any\_Other\_Name\_You\_Feel\_Like.)

What's going on is that if some client package says "with A.Types" and uses the Guid\_Write type, and uses Guid\_Write'Write(...) or Guid\_Write'Output(...), the client has to know that there's a Write routine that isn't the default. That's why the "for Guid\_Write'Write use ..." has to be in the visible part of A.Types, so that other clients are allowed to know about it.

*From: Adam Beneschan  
<adam@irvine.com>  
Date: Thu, 23 Jan 2014 15:43:00 -0800  
Subject: Re: Binary and XML serialization of types  
Newsgroups: comp.lang.ada*

> [...]

> But, more importantly, are you sure about this? GNAT's Ada.Containers.Vectors, for example, declares the stream-related stuff in the private part.

Sorry, I think I must have had caffeine deficiency syndrome when I wrote that.

You're right. Here's what I should have said:

What's going on is that "for Guid\_Write" specifies a property, or "aspect", of the type Guid, and that needs to be done in the same place where Guid is defined, which in this case is the package specification (although it could be in the private part).

Anyway, my goal was to help the OP understand intuitively why it wouldn't make sense to have a type declared in one package and then allow a property of that type to be changed in some other package. I hope I got that across, even if I did so badly.

*From: Randy Brukardt  
<randy@rrsoftware.com>  
Date: Thu, 23 Jan 2014 18:58:51 -0600  
Subject: Re: Binary and XML serialization of types  
Newsgroups: comp.lang.ada*

> [...]

To clarify (or confuse?) this more, it does have to be visible if Guid is a limited type, because in that case it won't have a usable 'Write unless it is explicitly defined. For other types, it can be in the private part.

### Untyped For Loops

*From: Adam Beneschan  
<adam@irvine.com>  
Date: Tue, 11 Feb 2014 15:56:52 -0800  
Subject: Re: character literals  
Newsgroups: comp.lang.ada*

[...]

This is different, though:

```
for Ch in '0' .. '9' loop
```

because this loop statement \*is\* the declaration of Ch, so the compiler has to be able to resolve the type just from the literals '0' and '9', and it can't. However, this is legal:

```
Start_Ch : Character;
for Ch in Start_Ch .. '9' loop
```

because now although '9' is ambiguous, the language will use the type of Start\_Ch to resolve the type of '9'. I don't think it's necessary (even from a style standpoint)

to include the type name in the "for" statement; others may differ.

The first "loop" statement, which is ambiguous, was legal in Ada 83, when there was only one character type; when `Wide_Character` was added to Ada 95 [`Wide_Wide_Character` wasn't added until Ada 2005], this became illegal, which caused some compatibility headaches for existing code.

Also:

```
type Traffic_Light is (Red, Yellow, Green);
type RGB is (Red, Green, Blue);
```

```
for Color in Red .. Green loop
-- ambiguous, illegal
for Color in Green .. Blue loop
-- legal, since there is only one meaning
-- of Blue
```

However, I'd definitely recommend including the type name in a case like this.

```
for Color in RGB range Green .. Blue loop
```

Finally, the language does have one special rule:

```
for I in 0 .. 9 loop
```

The literals 0 and 9 could be resolved to any integer type, which would make this ambiguous since there are normally multiple integer types visible in the program (`Integer`, `Long_Integer`, `Short_Integer`, maybe types in `Interfaces` if you "use" that packaged). But the language rules decree that the type will be `Integer` in that case. This is a situation where some programmers might recommend making the type `Integer` explicit.

*From: Robert A Duff  
<bobduff@shell01.TheWorld.com>  
Date: Wed, 12 Feb 2014 10:53:24 -0500  
Subject: Re: character literals  
Newsgroups: comp.lang.ada*

> [...]

Like me. I think the special-case for `Integer` is a kludge, so I would write:

```
for I in Some_Type range 0 .. 9 loop
```

One exception: If I want to say "do this 5 times", I might write:

```
for I in 1 .. 5 loop
```

and there are no references to `I` in the loop, so its type is irrelevant.

On the other hand, if the type is clear from the bounds, as in

```
for I in 1 .. Some_Array'Last - 1 loop
```

I wouldn't put the type in.

*From: Jean-Pierre Rosen  
<rosen@adalog.fr>  
Date: Wed, 12 Feb 2014 18:55:26 +0100  
Subject: Re: character literals  
Newsgroups: comp.lang.ada*

> [...]

```
>
> for I in Some_Type range 0 .. 9 loop
<shameless_plug> and this can be
enforced by AdaControl with the rule:
    check statements (untyped_for);
</shameless_plug>
[see-also "AdaControl", AUJ 34-3, p. 140.
—sparre]
```

## Locking Implementations

*From: Simon Belmont  
<sbelmont700@gmail.com>  
Date: Sat, 15 Feb 2014 12:20:00 -0800  
Subject: Implementation Locks  
Newsgroups: comp.lang.ada*

In the past, using other languages, I spent lots of time worrying about whether I should use a spinlock or a semaphore based on how long an sequence of operations was expected to be and how many CPU's were on the system, and wrote lots of overly complicated code to chose the best option in each situation.

Now in Ada, I have difficulty giving up the habit and lay awake at night worrying about whether the implementation is going busy-wait or block for a protected action or closed entry, especially now that everyone has multicore CPU's. I'm particularly consternated by closed entries, since I doubt the compiler can predict whether it will open back up in several microseconds or in several days. Is it unreasonable to expect an implementation to use some sort of dynamic, hybrid model that takes into account both how many CPU's are in the system and the average time to wait? Should I just trust the runtime and try not to worry? Is there even anything I can do about it either way?

*From: Jeffrey R. Carter  
<jrcarter@acm.org>  
Date: Sat, 15 Feb 2014 14:34:29 -0700  
Subject: Re: Implementation Locks  
Newsgroups: comp.lang.ada*

> [...]

You should trust the implementation, and measure your results. If you're meeting your timing requirements, then you have nothing to worry about.

## A Reminder on Compiler Warnings

*From: Robert A Duff  
<bobduff@shell01.TheWorld.com>  
Date: Sun, 16 Feb 2014 09:25:35 -0500  
Subject: Re: Best representation for spares  
Newsgroups: comp.lang.ada*

Pablo Rego <pvrego@gmail.com> writes:

```
> Ah, pragma Warnings (Off) is not fair.
When I get the warning, I prefer to
believe in the compiler. So it must exist
a better way to do it (and do not get the
warning).
```

You are wrong to always believe in the compiler. In GNAT, and most other compilers for Ada or any other language, a warning indicates that something MIGHT be wrong, not that something definitely IS wrong.

You are smarter than the compiler (although perhaps not as detail oriented). When you see a warning, you should inspect the code, and if you think the code is correct, use pragma `Warnings (Off, ...)` to suppress it, along with a comment explaining why. Don't write less-elegant code just to make the compiler shut up.

## Detecting Use of Unsafe Features

*From: Robert A Duff  
<bobduff@shell01.TheWorld.com>  
Date: Sun, 16 Feb 2014 09:13:26 -0500  
Subject: Re: Differences between Ada 83
and other revisions  
Newsgroups: comp.lang.ada*

Martin Dowie <martin@thedowies.com> writes:

```
> But at least it spells out that it is
potentially dangerous by being called
"Unchecked", like all the other
"Unchecked" parts of the language
...very easy to find!
```

If only that were true. I don't see any "unchecked" here:

```
for X'Address use ...;
```

```
X := ...;
```

It would be great if you could find all unsafe (i.e. potentially erroneous) code by searching for something like "unchecked". But alas.

On the bright side, Ada doesn't have very many unsafe features, and mostly allows them to be avoided and/or encapsulated. Compare with C, where every array indexing operation is unsafe.

*From: Jean-Pierre Rosen  
<rosen@adalog.fr>  
Date: Sun, 16 Feb 2014 16:58:24 +0100  
Subject: Re: Differences between Ada 83
and other revisions  
Newsgroups: comp.lang.ada*

> [...]

But `AdaControl` can find every use of (instantiations of) `Unchecked_*`, and all usages of `'Address`, or only address clauses that refer to the address of another object.

There needs to be a boundary between what is checked by the compiler and what is best handled by external tools; you may not agree to where the line has been drawn, but tools that can find unsafe features do exist!

[see-also "AdaControl", AUJ 34-3, p. 140. —sparre]

*From: Robert A Duff*  
*<bobduff@shell01.TheWorld.com>*  
*Date: Wed, 19 Feb 2014 17:09:31 -0500*  
*Subject: Re: Differences between Ada 83*  
*and other revisions*  
*Newsgroups: comp.lang.ada*

> [...]

That's useful. Can it find all unsafe features? There are some that are quite difficult to detect, such as passing a component of a variant record to a procedure that causes that component to vanish.

> [...]

I didn't mention any compiler checking up there. I said "search". I'm asking for a language-design principle that says "you cannot use any unsafe feature without with-ing a package called Unsafe, or a descendant thereof". Then a simple search for "unsafe" finds them all, without any need for sophisticated tools.

Can you name all the unsafe features of Ada off the top of your head, and tell what strings to search for to find them? I can't. You can find them by looking up "erroneous" in the Index.

(C is far worse in that regard!)

*From: Jean-Pierre Rosen*  
*<rosen@adalog.fr>*  
*Date: Wed, 19 Feb 2014 23:23:44 +0100*  
*Subject: Re: Differences between Ada 83*  
*and other revisions*  
*Newsgroups: comp.lang.ada*

> [...] all unsafe features? There are some that are quite difficult to detect, such as passing a component of a variant record to a procedure that causes that component to vanish.

Not this one, currently. But if you are willing to fund the development of this check, I'll be very happy to add it!

> [...]

Right, but be careful not to throw the baby with the bathwater. You can find many of the unsafe features, and that's much better than any other language!

## Discussion on a new Text I/O System

*From: Robert A Duff*  
*<bobduff@shell01.TheWorld.com>*  
*Date: Sun, 16 Feb 2014 11:17:16 -0500*  
*Subject: Re: Text\_IO, was: Re: Something I*  
*don't understand*  
*Newsgroups: comp.lang.ada*

> [...]

I/O should be separated from processing. In this case, the "processing" I'm talking about is formatting and parsing of data (e.g. turning an integer into a human-readable sequence of characters). See how it's done in Java.

Formatting for type T belongs with type T, not in Text\_IO.

Input should be separate from output. Put(Stream, X), where X is an integer makes sense, because we know X is an integer. Get(X) makes no sense, because we have no idea what the user is going to type. For text input, you need "read the next token, and tell me what it is", not "read an integer token, and blow up if the user typed something else".

A simplified and type-safe version of C's printf style (template-based) formatting would be more readable than concatenating a bunch of strings together to print messages, and MUCH better than using a series of Put calls to print a single message.

I/O should be task safe, at least for standard output and friends.

There are various ways operating systems have chosen to represent text files: lines separated by a single character, lines separated by two characters (CR/LF), record oriented. Obviously, the language design needs to pick one of those models, and the implementation needs to map that model onto whatever the OS does. Any model will work, but Ada chose the least convenient one.

Path names (file names with directory names and so on) should be represented using an appropriate type, with structure, properly interoperating with Ada.Directories. String is the wrong type for that. See how it's done in Common Lisp, quite portably.

The Get\_Line procedure invites people to write broken programs with arbitrary annoying line-length limitations. However long you make that String, it will be either too short or too long, and most likely both. The Get\_Line function is better.

There should be a convenient way to read an entire file into a String. Similar for writing.

String should be a private data type with appropriate operations, representing full Unicode, probably represented in UTF-8. But we can't blame the designers of Ada 83 for choosing 7-bit ASCII. Even that was a bold move, at a time when most languages didn't even define a standard character set, leaving it up to the operating system.

There should be a standard way to represent multi-line text in a String.

There is no convenient way to open a file in append mode, creating it (empty) if it doesn't exist, atomically.

Calling Create followed by Close, with no intervening output, does not create an empty file. That's broken.

The line-counting business is largely useless, and somewhat confusing.

The page-handling is largely pointless, and gets in the way even when you don't care about pages.

Finalization (which didn't exist in Ada 83) should be used to automatically close files

Open should be a build-in-place function (which didn't exist in Ada 83), instead of a procedure.

> [...] do you think Text\_IO should be outright replaced ?

You mean if compatibility were not a concern? Yeah, the only reason to keep Text\_IO as it is for compatibility. In a from-scratch language design, I'd do it rather differently.

*From: Simon Cluble*  
*<clubley@eisner.decus.org>*  
*Date: Mon, 17 Feb 2014 12:52:21 +0000*  
*Subject: Re: Text\_IO, was: Re: Something I*  
*don't understand*  
*Newsgroups: comp.lang.ada*

> [...]

>

> Formatting for type T belongs with type T, not in Text\_IO.

Agreed, but it should be a two-way thing.

There should be both External\_To\_Internal and Internal\_To\_External support to convert between the external (human readable) format and the internal format. You would also need to specify a format when going from internal to external format so the output would fit in the requested field width and obey the requested attributes (for example, number of decimal places).

[...]

*From: Georg Bauhaus*  
*<bauhaus@futureapps.de>*  
*Date: Mon, 17 Feb 2014 16:32:15 +0100*  
*Subject: Re: Text\_IO, was: Re: Something I*  
*don't understand*  
*Newsgroups: comp.lang.ada*

> A simplified and type-safe version of C's printf style (template-based) formatting [...]

I wonder if this feature could be tacked onto the string types? With the help of attribute functions and named bindings, formatting could be handled flexibly, leaving room for internationalization, for example. Formatting could also be handled conveniently, insofar as the language provides "obvious" default formatting.

```
For_Invoice : constant Wide_String :=
  -- not Ada
  Wide_String'Edited ("A total of %{Sum}
    %{Currency} for %{Pieces}")
with Bindings =>
  (Sum => (Value => <>,
    Money'Wide_Formatted =>
      Using_Pic_String),
  Currency => (18N.CU,
    Money'Wide_Formatted => <>),
  Pieces => (Amount * 2.0,
    Three_Columns'Access));
Format : constant -- needs to be a static
  -- constant
```

```
Wide_String'Edited := "Process #
                        %{pnum} : %{n}µs";
```

A  $\langle \rangle$  stands for the default choices, viz. a variable or parameterless function named "Sum" in the first row, and a default "formatter" in the third. When no specialized formatting is needed for an item, write

```
with Bindings =>
  (...
   Foo =>  $\langle \rangle$ ,
   ...);
```

As an example of a flexible solution, "Using\_Pic\_string" from the first example above would be a function with a profile like that of "Wide\_Image. (Its body uses existing language features, in this case picture strings from Information Systems Annex.) Also, since the generics of Text\_IO already provide for formatting numbers (Putting them into strings), these routines could be borrowed for the meaning of

```
T'[Wide_][Wide_]Formatted
```

Safety of the template is guaranteed insofar as the number of items (and types, I think) in any actual binding is known at compile time; type checking looks possible. Hence, the simplest formatting would be

```
String'Edited("%{n} bottles of beer on the
wall")
with Bindings (others =>  $\langle \rangle$ );
```

It requires only that there be a variable/function named N.

From: Niklas Holsti  
<niklas.holsti@tidorum.fi>  
Date: Mon, 17 Feb 2014 18:59:43 +0200  
Subject: Re: Text\_IO, was: Re: Something I don't understand  
Newsgroups: comp.lang.ada

> A simplified and type-safe version of C's printf style (template-based) formatting [...]

I disagree, but then I don't understand how Robert would make C's template idea type-safe -- might Robert expand on his ideas?

I think that the present method of concatenating strings or using several Puts is good; what is needed is to extend or replace the 'Image attribute with similar value-to-string functions which are more controllable, flexible, and work also for composite types. Perhaps something analogous to the stream attributes, but with the ability to control the output format at each invocation, which is not possible with the stream attributes.

From: Dmitry A. Kazakov  
<mailbox@dmitry-kazakov.de>  
Date: Mon, 17 Feb 2014 18:17:16 +0100  
Subject: Re: Text\_IO, was: Re: Something I don't understand  
Newsgroups: comp.lang.ada

> [...] to extend or replace the 'Image attribute with similar value-to-string functions which are more controllable, flexible, and work also for composite types.

Except that all these need to be MD primitive operations. There is no way to solve this without MD.

Needless to say that templates could solve nothing only add further problems.

> Perhaps something analogous to the stream attributes, but with the ability to control the output format at each invocation, which is not possible with the stream attributes.

I don't think there is any need in having formats. A few formatting parameters could be passed along to Image or equivalent.

Environment settings (e.g. locale) should come from the rendering surface object. No need to specify them at all. This is how stuff like fonts, colors etc is handled in GUI.

From: Niklas Holsti  
<niklas.holsti@tidorum.fi>  
Date: Mon, 17 Feb 2014 19:42:07 +0200  
Subject: Re: Text\_IO, was: Re: Something I don't understand  
Newsgroups: comp.lang.ada

> [...]

Why multiple dispatch? Which would be the multiple controlling parameters? I think only the input value should be controlling; perhaps you think that the output channel/device should also be controlling?

> [...] A few formatting parameters could be passed along to Image or equivalent.

Well, parameters and options is what I meant. For example, the ability to specify blank fill, zero fill, center/left/right alignment, digit group spacing (1 123 456,00 or 1\_123\_456.00), etc.

> Environment settings (e.g. locale) should come from the rendering surface [...]

A "rendering surface" is not always available at the point where the string is generated.

There could be a private predefined type for such settings. A value of that type could be given as a parameter in the Image call to set the default format (which could then be overridden if the Image call also has some specific format parameters). A GUI toolkit could have a function to return a suitable value of this type from a "rendering surface" object. I don't see why this, or the output channel, should be a controlling parameter.

Something like these flexible Image functions already exists in Annex F (Information Systems), but it is partly template-driven ("picture"-driven).

From: Dmitry A. Kazakov  
<mailbox@dmitry-kazakov.de>  
Date: Mon, 17 Feb 2014 20:55:12 +0100  
Subject: Re: Text\_IO, was: Re: Something I don't understand  
Newsgroups: comp.lang.ada

> Why multiple dispatch?

Print (Display, Shape)

is a textbook example of MD.

> Which would be the multiple controlling parameters?

File and Value

> [...] perhaps you think that the output channel/device should also be controlling?

Certainly so. Consider ASCII\_File, UTF8\_File, Gtk\_Text\_Buffer\_Record and so on. You cannot convert to string before sending it out, because ASCII will use E for power of 10, UTF8 will use superscript characters for it, and Gtk\_Text\_Buffer\_Record will do the GTK markup language.

[...]

> A "rendering surface" [...]

String itself is such a surface. That is the point of having it controlled. You can replace it with whatever type, e.g. File, Stream etc. And string itself is a Universe of types because of encodings.

> There could be a private predefined type for such settings. A value of that type could be given as a parameter in the Image call to set the default format (which could then be overridden if the Image call also has some specific format parameters).

Yes of course, but this is usually another set of parameters. Less volatile parameters, such as whether to use '!' or '!' to introduce fraction belong to the surface.

> A GUI toolkit could have a function to return a suitable value of this type from a "rendering surface" object.

Possible but tedious.

> I don't see why this, or the output channel, should be a controlling parameter.

Because you cannot predict all possible combinations of and because it is a huge wasting of human and computational resources as no given application will ever use more than 1% of it.

From: Niklas Holsti  
<niklas.holsti@tidorum.fi>  
Date: Tue, 18 Feb 2014 09:14:26 +0200  
Subject: Re: Text\_IO, was: Re: Something I don't understand  
Newsgroups: comp.lang.ada

> [...]

> Certainly so. Consider ASCII\_File, UTF8\_File, Gtk\_Text\_Buffer\_Record and so on.

That could be done using overloading based on the expected type of the Image function result, instead of multiple dispatch. Of course that would require different types to represent ASCII strings and UTF8 strings, etc. Using different types for different kinds of strings would be a good thing anyway, IMO (but I know that this causes problems with a combinatorial explosion of the number of predefined subprograms involving strings).

> You cannot convert to string before sending it out [...]

Hrm. I'm not at all sure that I would want such different formatting for different output channels to happen automatically. For one thing, using superscripts for exponents would prevent or complicate the user's copy-paste operations, for example copying the output of a Float number into a calculator accessory.

There should really be a type "Text" that represents text, with all its complications of encoding, formatting, styles, fonts, lines, paragraphs, tabulation, indentation, language, ... True, that is horribly complex, but that's the reality now. It is debatable if this should be in the language, or in toolkits (GUI or others). Probably some core part of it should be in the language and the rest in a toolkit or in an optional Annex to the language.

From: Niklas Holsti  
<niklas.holsti@tidorum.fi>

Date: Wed, 19 Feb 2014 10:36:29 +0200  
Subject: Re: Text\_IO, was: Re: Something I don't understand  
Newsgroups: comp.lang.ada

> [...] You mean text buffer like in GUI?

Possibly so, but I'm not familiar with GUI text buffers. It sounds like a similar thing.

> To me text buffer, stream, file, string are all instances of the class of types over which Put dispatches. OK, we can call the abstract root type of the class "Text."

Perhaps Put would just be overloaded for these types. It depends if you consider the types a class, or not. I'm not sure what is best.

> Yes, that Put would be an implementation of the primitive operation defined for the class Text.

I would make Text a type, not a class. This would avoid the need for multiple dispatch on a controlling Text parameter.

I'm thinking of two levels of "Put":

```
Put (To : in out Text, Item : in String);
```

Add items to a Text, building a logically structured Text, but without rendering it yet. This will probably need some concept of "points in a Text where more stuff can be inserted" so that the Put can preserve or extend the logical Text structure.

```
Put (To : in out Text, Item : in Text);
```

Render the Text into some external File.

The Text buffer intermediary means that each level of Put can (if desired) be dispatching on one of the parameters, without needing multiple dispatch.

> I think it is only logical for a strongly typed language to map this kind of stuff onto types.

I agree.

> This search for other "ways" (aspects, generics etc) is really damaging the language.

I don't agree. I see aspects as strengthening or broadening the type concept, and generics as a meta-type level. But I must admit that I have lately been using generics less often and have instead used classes and dispatching more often. However, I think that generics are useful and are not entirely subsumed by classes.

From: Niklas Holsti  
<niklas.holsti@tidorum.fi>

Date: Wed, 19 Feb 2014 15:20:18 +0200  
Subject: Re: Text\_IO, was: Re: Something I don't understand  
Newsgroups: comp.lang.ada

> [...]

My notion of type "Text" is an internal representation of text meant for human reading and viewing. I don't see any logical need for making this type a class; there would be only one predefined (and private) type.

(There might be some technical reasons for making the type a class, for example with a root type that is low-cost but simple, and some derived classes which provide more functionality but at greater cost. Perhaps the simple root type would be a mandatory language feature, and the derived classes optional features.)

By the way, perhaps the word "text" is ambiguous. I think it is time to make a clear distinction between:

(1) a text file (sometimes called an "ASCII file"), which is a sequence of basic symbols (e.g. Character or Wide\_Character) used to represent \*data\* for either reading by another program, or for human reading (without formatting), and

(2) a text meant only for human reading/viewing and therefore to be rendered as nicely and readably as the chosen viewing device allows. That some parts of the text can be seen as sequences of characters is secondary, and the specific characters and their sequence can change according to the rendering.

Ada.Text\_IO implements mainly (1), with some basic support for typewriter-style formatting (column spacing, line spacing, page tracking).

The "Text" type I am talking about aims to be the internal representation of (2), before rendering on some viewing device.

> [...] This was attempted before, many many times, actually. From PostScript to HTML, an intermediate language that would take care of separating higher level formatting from lower level rendering. It never worked how many times tried.

Uh... surely PostScript and HTML "work". I'm pretty sure that a large fraction, perhaps even a majority of programs today generate most of their human-readable output as HTML. Even if the final HTML generation is delegated to some web-application framework.

> And for sure, it will be even more hated than Text\_IO page formatting is, because the overhead will be far bigger. Imagine describing the semantics of, say, conversion of File, Stream, String to Text and backward.

Overhead compared to what? If the need is to generate nicely formatted output, rendered in device-specific ways, and typewriter formatting is not enough, what is the alternative?

The overhead of Text\_IO are important only when processing large text \*data\* files (meaning (1) of "text"). For generating human-readable text (meaning (2)), especially in an interactive context, the overhead is utterly negligible.

I don't see any need for converting a File/Stream \*into\* Text, unless the File/Stream is a serialized representation of the full internal structure of a Text object, in which case the File/Stream structure is private and normal serialization/deserialization methods apply.

I don't intend that the type "Text" should be so fancy and complete that it could be used as such to implement an advanced word processor. Following the same rationale as Ada.Containers, "Text" should provide as much functionality as can be expected to be useful for (and used by) many Ada programs and programmers, but programmers requiring high performance or high/specific functionality would have to implement more advanced "text" representations themselves.

From: Robert A Duff  
<bobduff@shell01.TheWorld.com>  
Date: Wed, 19 Feb 2014 16:46:45 -0500  
Subject: Re: Text\_IO, was: Re: Something I don't understand  
Newsgroups: comp.lang.ada

> [...] how Robert would make C's template idea type-safe -- might Bob expand on his ideas?

My answer depends on if (or how much) I'm allowed to change Ada's type system. I've done this in pure standard Ada. Something like:

```
type Template is new String;
procedure Put (T : Template; X1, X2, X3,
              X4, X5, X6, X7, X8 : String := "");
```

```
Put ("There were \1 warnings and \2
errors.\n",
Image (Warning_Count),
Image (Error_Count));
```

prints the template as is to standard output, except it replaces \1 with X1, and \2 with X2, and the "\n" is new-line. You can have multi-line templates. "\\ represents \".

Note my use of X1...X8 to simulate C's variable-length parameter lists. This is a kludge.

Note that Template is a different type from String. This prevents a bug that can happen in C, where you say printf(blah), and blah is some data read off the internet. That can be a security hole!

The user is responsible for writing suitable Image functions for their data types, and they can take whatever formatting parameters you like. This seems much more readable than C's way of encoding the field widths and whatnot in the template.

For localization/internationalization, you can have a table mapping "There were \1 warnings and \2 errors.\n" to the corresponding template in (say) French. Different languages have different word orders, so you might have:

```
"... \2 ... \1 ... \n"
```

to reverse the order of insertion.

This is all 100% type safe. Most of the checking is static. It checks at run time that the number of \1, \2, \3, ... escapes matches the number of non-empty Xn parameters passed.

Now, if you let me change Ada, I'd allow user-defined literals. Any type derived from the Has\_Literals interface allows literal syntax, and it overrides the Literal\_Value function to convert the sequence of characters to that type. Template would no longer need to be derived from String; it could be a private extension of Has\_Literals, and the Literal\_Value function could "precompile" the template into some convenient/efficient internal form.

The Literal\_Value call is evaluated at compile time, so the above-mentioned run-time check can now be static.

Change the types of the Xn parameters to Has\_Image'Class, so you can pass Warning\_Count, and it automatically dispatches to Image(Warning\_Count). If you want to use extra formatting options you'd call your own Image function explicitly. Integer types are derived from Has\_Image, and the Image function is not an attribute, and (most importantly of all!) it doesn't insert an annoying extra blank.

I'd also allow variable-length argument lists and/or arrays of strings.

> I think that the present method of concatenating strings or using several Puts is good;

I think Put (at least to standard output/error) should be task safe. That is, it should be atomic with respect to other tasks doing Put. Puts from different tasks would be interspersed, but you wouldn't get character-by-character interspersal, and you certainly wouldn't get the Ada rule ("erroneous and therefore unpredictable behavior").

That rules out the "series of Puts" method. You need to build up your whole message (possibly multi-line) and then Put it in one fell swoop.

The concatenating strings method just looks ugly to me -- I can't easily see what the message is going to look like. With a template, I see the whole message, with marks for where variable data is inserted.

[...]

From: Niklas Holsti

<niklas.holsti@tidorum.fi>

Date: Thu, 20 Feb 2014 22:02:42 +0200

Subject: Re: Text\_IO, was: Re: Something I don't understand

Newsgroups: comp.lang.ada

> [...] Note my use of X1...X8 to simulate C's variable-length parameter lists. This is a kludge.

That is where I think this breaks down for current Ada. To really add this to Ada, you need variable-length parameter lists, or the ability to aggregate strings of various lengths into a vector of strings.

> [...]

But Template objects can still be variable, so they can still be constructed at run-time from external data (say, some application configuration file). Or do you intend to define Template as a type that forbids variable objects and allows only static constants?

But even with variable Templates, this is clearly less risky than the case in C, where AIUI the risk comes from malicious type breaking where the format conversion character misuses the corresponding parameter. That cannot happen in your proposal because all parameters are Strings and there are no format conversion characters in the Template.

[...]

> Now, if you let me change Ada,

Apart from the type-safe variable-length parameter lists, you mean? :-)

> I'd allow user-defined literals. [...]

Not a bad idea. Would only string literals qualify, not character or numeric ones? To avoid ambiguities, this should be forbidden for any type that already allows string literals, right?

But this is beginning to look a bit like C++ parameterized constructors, called implicitly... slippery slope?

> [...] The Literal\_Value call is evaluated at compile time, so the above-mentioned run-time check can now be static.

If the Template is defined by a literal, yes. But I assume Templates could still be variable objects, too, forcing a run-time check.

[...]

From: Robert A Duff

<bobduff@shell01.TheWorld.com>

Date: Wed, 19 Feb 2014 16:15:10 -0500

Subject: Re: Text\_IO, was: Re: Something I don't understand

Newsgroups: comp.lang.ada

[... The] "read the next token, and tell me what it is" method is how it should normally be done, and if the language

doesn't support it, then the programmer will do that, which is fine.

But the "read an integer token, and blow up if the user typed something else" method is rarely useful. I'd stick with "read a character", "read a line", and "read the whole file".

> [...]

> [Reading an entire file:] One for binary (unchanged) input and one for text input with end of line conversions.

Maybe. Should the binary one return a String, or an array of bytes, or something else?

From: Simon Clubleby

<clubley@eisner.decus.org>

Date: Wed, 19 Feb 2014 22:01:43 +0000

Subject: Re: Text\_IO, was: Re: Something I don't understand

Newsgroups: comp.lang.ada

[...]

> Maybe. Should the binary one return a String, or an array of bytes, or something else?

Conceptually, I would have to say an array of bytes data type distinct from a String. This is based on my perception of a String as been associated with text data (after all, it `_is_` called a String :-)) and that conceptually you cannot really assign such meaning to something you have gone to the trouble to read as binary data.

What really matters is that you can guarantee the binary data will not be altered during the I/O process, that you know the exact length of the data, and that you can cleanly access the bytes in the binary data.

From: Georg Bauhaus

<bauhaus@futureapps.de>

Date: Tue, 18 Feb 2014 09:04:27 +0100

Subject: Re: Text\_IO, was: Re: Something I don't understand

Newsgroups: comp.lang.ada

> [...]

Starting from some frequent use cases, such as

- logging and similar technical status reports,
- invoices, bank statements and other economic reports

static constant formats would cover a lot. I'd think that generics, as Mark H noted, might allow for elaborating minor variations at run-time, at least in some cases.

A big plus of templates is that they are instances of Bentley's programming pearls, of proven value. They shift the focus from how the language achieves printing to what is being printed, the latter being what matters.

## Recursive Type Invariants

*From: Anh Vo <anhvofraus@gmail.com>  
Date: Tue, 25 Feb 2014 19:29:45 -0800  
Subject: Class Wide Type Invariants - My bug or compiler bug  
Newsgroups: comp.lang.ada*

GNAT did not raise `Assertion_Error` where I thought it should for the following codes. Either I misunderstood the LRM or it is a compiler bug.

```
package Places is
  type Disc_Pt is tagged private
    with Type_Invariant'Class =>
      Check_In (Disc_Pt);
  Initial_Disc_Pt : constant Disc_Pt;

  function Check_In (D : Disc_Pt)
    return Boolean with Inline;

  procedure Set_X_Coord
    (D : in out Disc_Pt; X : Float)
    with Pre => (X >= -1.0 and then
      X <= 1.0);

  procedure Set_Y_Coord
    (D : in out Disc_Pt; Y : Float)
    with Pre => (Y >= -1.0 and then
      Y <= 1.0);

private
  type Disc_Pt is tagged
    record
      X, Y : Float range -1.0 .. +1.0;
    end record;

  Initial_Disc_Pt : constant Disc_Pt :=
    (others => 0.5);
end Places;

package body Places is

  function Check_In (D : Disc_Pt)
    return Boolean is
  begin
    return (D.X**2 + D.Y**2 <= 1.0);
  end Check_In;

  procedure Set_X_Coord
    (D : in out Disc_Pt; X : Float)
```

```
begin
  D.X := X;
end Set_X_Coord;

procedure Set_Y_Coord
  (D : in out Disc_Pt; Y : Float)
begin
  D.Y := Y;
end Set_Y_Coord;
end Places;

package Places.Inner is
  type Ring_Pt is new Disc_Pt with
  private
    with Type_Invariant'Class =>
      Check_Out(Ring_Pt);

  Initial_Ring_Pt : constant Ring_Pt;

  function Check_Out (R : Ring_Pt)
    return Boolean
    with Inline;
private
  type Ring_Pt is new Disc_Pt with null
  record;

  Initial_Ring_Pt : constant Ring_Pt :=
    Ring_Pt'(Initial_Disc_Pt
      with null record);

  function Check_Out (R : Ring_Pt)
    return Boolean is
      (R.X**2 + R.Y**2 >= 0.25);
  end Places.Inner;

  with Ada.Text_IO;
  with Ada.Exceptions; use Ada;

  with Places.Inner;

  procedure Invariants_Inheritance_Test is
    use Text_IO;

    Child_Pt : Places.Inner.Ring_Pt :=
      Places.Inner.Initial_Ring_Pt;
  begin
    Places.Inner.Set_X_Coord(Child_Pt, 0.0);
    -- OK since 0.5**2 + 0.0 >= 0.25
    Places.Inner.Set_Y_Coord(Child_Pt, 0.1);
    -- should fail Check_Out(...),
    -- 0.1**2 + 0.0 < 0.25
  exception
    when Err : others =>
      Put_Line ("Houston help!!! " &
        Exceptions.Exception_Information(Err));
  end Invariants_Inheritance_Test;

From: Adam Beneschan  
<adam@irvine.com>  
Date: Wed, 26 Feb 2014 14:35:52 -0800  
Subject: Re: Class Wide Type Invariants -  
My bug or compiler bug  
Newsgroups: comp.lang.ada
> [...]
```

It looks to me like this should work, according to 7.3.2(19). I don't know what GNAT's default `Assertion_Policy` for `Type_Invariant'Class` is, however.

*From: Randy Brukardt  
<randy@rrsoftware.com>  
Date: Wed, 26 Feb 2014 19:01:45 -0600  
Subject: Re: Class Wide Type Invariants -  
My bug or compiler bug  
Newsgroups: comp.lang.ada*

> [...]

7.3.2(19/3) is a mess, however. AI12-0042-1 changed it a lot, but that change isn't right either, so it's rather in limbo at the moment. (See the working RM for the current state of things.)

Note that a literal implementation of 7.3.2(19/3) would cause every invariant check to go infinitely recursive, since there is supposed to be an invariant check on the parameter of `Check_In`, which is called from the invariant check - repeat forever. GNAT doesn't implement that for obvious reasons, so it can't exactly implement the rule as written, and once you have to go off the grid, all bets are off.

Some parts will be in every rule (checking of in out and out parameters, for instance), so you probably can assume those are checked. But that's about it. Probably it would be better to not depend too much on `Type_Invariants` until we figure out what rules actually make sense (and we find a set that isn't insane for one reason or another).

## Request from the ARG: Static Constants

*From: Randy Brukardt  
<randy@rrsoftware.com>  
Date: Mon, 3 Mar 2014 18:03:56 -0600  
Subject: Request for help from the ARG:  
Static constants  
Newsgroups: comp.lang.ada*

There is a language-lawyer level question that some of the ARG members have been discussing privately for far too long in the past week. An important part of the question is the compatibility effect of changing the rules to match the expectation. As such, I'd like to find out what various compilers do on the following test program. I've tried recent versions of GNAT and Janus/Ada, both of which reject the program citing an error at (2). [This is not supported by the RM wording, BTW.] If you have access to some other Ada compiler, please attempt to compile this program and report the result, either here or to me privately (randy@rrsoftware.com).

```
with Ada.Text_IO;
procedure SC is
  Item_Size : constant := 0;
begin
  Ada.Text_IO.Put_Line ("Start Static
    Constant check");
  if Item_Size > 0 then
    declare
      Length : constant Positive :=
        Item_Size; -- (1)
```

```

type Data_Index is range 1 .. Length;
-- (2)
type Data_Array is array (Data_Index)
of Natural;
begin
  Ada.Text_IO.Put_Line
    ("Can't get here");
exception
  when Constraint_Error =>
    Ada.Text_IO.Put_Line
      ("Can't get here, either");
end;
else -- Do nothing
  Ada.Text_IO.Put_Line
    ("Nothing as expected");
end if;
Ada.Text_IO.Put_Line
  ("End Static Constant check.");
end SC;

```

From: Tom Moran <tmoran@acm.org>  
 Date: Tue, 4 Mar 2014 02:59:46 +0000  
 Subject: Re: Request for help from the ARG:  
 Static constants  
 Newsgroups: comp.lang.ada

> (2). [This is not supported by the RM wording, BTW.] If you have access to some other Ada compiler, please attempt to compile this program and report

An ancient ObjectAda compiler gives:

```
-----Target: Win32 (Intel)
Debug-----
```

```
sc.adb: Warning: line 8 col 39
LRM:11.5(17), Value outside range,
Constraint_Error will be raised
```

```
Front end of f:\oa722\console\sc.adb
succeeded with no errors.
```

```
Tool execution has completed.
```

## Checking “out” Parameters

From: Tero Koskinen  
 <tero.koskinen@iki.fi>  
 Date: Sat Mar 8 2014  
 Subject: Checking "out" parameters with  
 Adacontrol  
 URL: <http://ada.tips/checking-out-parameters-with-adacontrol.html>

Have you ever accidentally written code like this?

```

procedure Example_Proc (X : out
                        Boolean) is
begin
  null; -- Do something, but do not touch X
end Example_Proc;

with Example_Proc;
procedure Main is
  My_Flag : Boolean;
begin
  Example_Proc (My_Flag);
end Main;

```

In the above code, parameter X with mode “out” is left untouched. Because of this, value of My\_Flag is undefined after Example\_Proc (My\_Flag) call.

To prevent mistakes like this, you can use Adacontrol and a rule:

```

check improper_initialization
(out_parameter);

```

With the rule, Adacontrol will warn you about your mistake:

```

$ daactl -f rules.aru example_proc.adb
example_proc.ads main.adb

```

```

example_proc.adb:1:25: Error:
IMPROPER_INITIALIZATION: out
parameter "X" not safely initialized

```

```

$
[see-also "AdaControl", AUJ 34-3, p. 140.
—sparre]

```

## Simplifying the Language?

From: Randy Brukardt  
 <randy@rrsoftware.com>  
 Date: Thu, 20 Mar 2014 18:15:30 -0500  
 Subject: Re: Augusta: An open source Ada  
 2012 compiler (someday?)  
 Newsgroups: comp.lang.ada

> [...]

Heck, we (the ARG) aren't quite sure how you implement accessibility checks for Ada 2005 and Ada 2012 (see AI12-0016-1 for some thinking); you could waste a lot of time trying to figure that out. And like J-P says, a 95% solution isn't good for much -- the real solution is 95% different. :-)

It's for good reason that 3.10.2 is informally named "The Heart of Darkness"! ;-)

From: J. Kimball <jkimball4@gmail.com>  
 Date: Mon, 24 Mar 2014 03:18:21 -0500  
 Subject: Re: Augusta: An open source Ada  
 2012 compiler (someday?)  
 Newsgroups: comp.lang.ada

> [...]

It's becoming abundantly clear that there has to be a massive break in backward compatibility in the next revision of the language that makes writing compilers easier, not just keeping AdaCore in business, but breaking out of the framework of Ada 95.

We find ourselves discussing this regularly in #ada on Freenode. Many of us see Ada as a sinking ship because of all its baggage. The ideals are strong, but the implementation is losing us.

I surely need to review the AIs for the next revision to see what's happening.

From: Peter C. Chapin  
 <PChapin@vtc.vsc.edu>  
 Date: Mon, 24 Mar 2014 08:51:00 -0400  
 Subject: Re: Augusta: An open source Ada  
 2012 compiler (someday?)  
 Newsgroups: comp.lang.ada

> [...]

This is one reason why having multiple implementations is a good thing. As an example the C++ community basically

decided that template export, as required by the C++ 1998 standard, wasn't worth the implementation difficulties. As a result export has been removed from the C++ 2011 standard... despite the fact that there was one (only one) compiler that implemented it.

If another compiler existed that \*almost\* implemented Ada 2012 but left out controversial features (are there any?), and if that compiler proved acceptable and useful to a significant part of the community, it would help provide a kind of reality check on the standardization process.

I'm not saying Augusta will ever be mature enough to do this. I'm speaking here in general terms about the value to the community of having multiple competing implementations. It certainly seems, at the moment, as if GNAT is the only viable Ada 2012 compiler in existence and that isn't healthy for Ada.

From: Randy Brukardt  
 <randy@rrsoftware.com>  
 Date: Mon, 24 Mar 2014 16:21:20 -0500  
 Subject: Re: Augusta: An open source Ada  
 2012 compiler (someday?)  
 Newsgroups: comp.lang.ada

> [...] Your own compiler can't even compete because of the labyrinth of rules.

Not really. My compiler can't compete because I'm a lousy businessman and to a lesser extent because I'm rather burned out. Some of these corner cases (especially "the Heart of Darkness") are obscure corners of the language of little interest to anyone. Until you try to get rid of them, and then the safety argument rears up (dangling pointers are a scourge).

> [...] It's becoming abundantly clear that there has to be a massive break in backward compatibility in the next revision of the language that makes writing compilers easier [...]

I'd be in favour of that, but I'm dubious that the customers that support Ada would want to make that sort of change. And if the customers don't come along, then there is little energy for anything to happen. After all, most hobbyist driven projects tend to wane after a couple of years, and that's not going to work for the sorts of long-lived projects that Ada is best at.

> [...] The ideals are strong, but the implementation is losing us.

I could see some relatively small tweaks, but I doubt that would help implementation effort much. In particular, one of the nastiest things is type resolution. But the part of type resolution that is hard is the ability to overload on result types. (That's not allowed by C++, for instance.) But a large part of the elegance and ease-of-use of operators comes from that ability. Taking it away would prevent a lot of common



techniques (for instance, it's what allows overloading of enumeration literals).

I'd be more interested in regularizing some of the rules (such as making objects overloadable) -- but I doubt that would have any positive impact on the effort to implement Ada.

One could try removing/altering large but not frequently used areas -- fixed point, tasks, discriminant-dependent components come to mind -- but for each one, you'd lose a bunch of Ada fans.

> [... next revision ...]

Nothing to speak of yet; too soon after Ada 2012 to do anything formally. We're just gathering ideas at this point.

*From: Dennis Lee Bieber*

*<wlfraed@ix.netcom.com>*

*Date: Mon, 24 Mar 2014 19:18:16 -0400*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [...] I'd be in favor of that, but I'm dubious that the customers that support Ada would want to make that sort of change. [...]

Aye; The paying customers aren't going to put up with the cost of recertifying something like a flight management system because a language revision has dropped support for some feature (or just made a small change in the semantics of existing syntax). And such systems may be in use for 20+ years.

I think I've overheard stuff at work where they are talking about having to do side-by-side examination of the generated object code to validate a new release of the compiler -- without changing the language standard in use.

*From: J. Kimball <jkimball4@gmail.com>*

*Date: Mon, 24 Mar 2014 18:50:18 -0500*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [...]

Why would an ATCS system change language revision at all? Anyone whose using features they don't want to give up can safely stay with old revisions of the language. GNAT has had those -gnat{83,95,05,12} switches for a long time. These are not valid reasons for not shaking things up. Even if some project decided to leave Ada, you may just as easily find new people approaching the language. Large projects who think just changing the switch in their Makefile to the new language revision is sufficient probably shouldn't be using Ada in the first place.

*From: Stefan Lucks*

*<stefan.lucks@uni-weimar.de>*

*Date: Tue, 25 Mar 2014 10:37:55 +0100*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [...]

What about moving not-so-often used language features into an annex?

Thus, if your customers demand the feature, you are allowed to support it. Furthermore, anyone supporting that feature would do so in a completely compatible way.

But if you don't want to support that feature, or you can't for some reason, you are allowed to support Ada 20XY without that annex.

As an example, I would consider interfaces. The support for multiple inheritance from "interface" could be moved into an annex, and thus become optional for the language implementer. The keyword "interface" should remain reserved, for compatibility reasons.

*From: Randy Brukardt*

*<randy@rrsoftware.com>*

*Date: Tue, 25 Mar 2014 15:47:35 -0500*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [...]

This would work, but it wouldn't have any effect on simplifying the Standard (since the wording in the Standard would have to be prepared to handle the feature -- that's especially likely to be an issue for things like interfaces, which have an effect on virtually every definition in the Standard). And if one doesn't simplify the Standard, it's fairly unlikely that you're actually going to simplify implementation that much.

> [Moving interfaces to an annex.]

Right.

But as always, the difficulty would be agreeing on what goes into such an annex. I'd vote for interfaces and anonymous access types, but I'm sure the fans of those features would not be very happy. And they probably have some features that I find important that they think ought to be in the junk bin.

You'd be amazed at how hard it is to even move a feature into Annex J (Obsolete Features), even those are required to be implemented. I'm still hearing flack about the decision to move aspect pragmas there, even though entity-specific pragmas was one of worst ideas ever known to mankind. :-)

*From: Michael B.*

*Date: Tue, 25 Mar 2014 20:41:23 +0100*

*Subject: Re: Augusta: An open source Ada 2012 compiler (someday?)*

*Newsgroups: comp.lang.ada*

> [...] It's becoming abundantly clear that there has to be a massive break in backward compatibility in the next revision of the language that makes writing compilers easier [...]

But breaking compatibility is very dangerous. Python did this and now there are two incompatible languages: Python 2.x and Python 3.x. Many library maintainers said, they will never support 3.x. Pascal made the same mistake. Instead of enhancing the language, Modula and Oberon were created. Today none of these three languages is used anywhere. I'm sure Ada would suffer a similar fate.

## On the Value of Interfaces and Multiple Inheritance

*From: Randy Brukardt*

*<randy@rrsoftware.com>*

*Date: Fri, 21 Mar 2014 18:02:28 -0500*

*Subject: Re: How to hide inherited implementation of a public interface?*

*Newsgroups: comp.lang.ada*

Exactly. "Abstract types" (which can have components, implementations, etc.) are not worthless. The restrictions on interfaces make them worth little. ("Worthless" is going a bit far, of course, but it makes a good sound bite.) The costs of multiple inheritance (which are considerable) make them not worth the effort.

Full multiple inheritance CAN be implemented, but it's expensive enough in compiler and language complexity that the costs outweigh the value. The halfway Java-like solution is easier to implement but makes no one happy. We should have told the multiple inheritance nuts to forget it, because it makes no sense for Ada.

*From: Dmitry A. Kazakov*

*<mailbox@dmitry-kazakov.de>*

*Date: Sat, 22 Mar 2014 09:31:56 +0100*

*Subject: Re: How to hide inherited implementation of a public interface?*

*Newsgroups: comp.lang.ada*

> [...]

In the real-life project I am working on, lack of proper MI led to a massive cut-and-paste code explosion on the scale 1 to 1000, at least.

> [...]

MI is not a language property, it is more of software design. You cannot get rid of the fact that software engineers will keep on trying to reuse code, sharing the code implementing file reading in the code of read-only and read-write files. This is a sound design. It is the opposite [\*] that does not make sense. If the language does not support sound software design decisions, well, that is what we call a language flaw.

[\*] The standard library is full of flaws caused by not using MI. From minor issues that `Root_Stream_Type` does not implement `Limited_Controlled`, to massive mess that `Character` and `Wide_Character` to don't share common interface.

# 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.

## 2014

- July 21-25      **38th Annual International Computer Software and Applications Conference (COMPSAC'2014)**, Västerås, Sweden. Topics include: software engineering, security and privacy, quality assurance and assessment, embedded and cyber-physical environments, etc.
- July 21-25      **10th European Conference on Modelling Foundations and Applications (ECMFA'2014)**, York, UK. Topics include: domain specific modelling languages and language workbenches; model reasoning, testing and validation; model transformation, code generation and reverse engineering; Model-Based Engineering (MBE) environments and tool chains; MBE for large and complex industrial systems; MBE for safety-critical systems; comparative studies of MBE methods and tools; etc.
- July 21-25      **4th International Workshop on New Algorithms and Programming Models for the Manycore Era (APMM'2014)**, Bologna, Italy. Topics include: parallelisation with appropriate programming models and tool support for multi-core and hybrid platforms; software engineering, code optimisation, and code generation strategies for parallel systems with multi-core processors; etc.
- July 21-25      **Software Technologies: Applications and Foundations (STAF'2014)**, York, UK. Successor of the TOOLS federated events. Topics include: practical and foundational advances in software technology, from object-oriented design, testing, mathematical approaches to modelling and verification, transformation, model-driven engineering, aspect-oriented techniques, and tools.
- ☺ Jul 28 - Aug 08      **28th European Conference on Object-Oriented Programming (ECOOP'2014)**, Uppsala, Sweden. 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 and design patterns, aspects, components, modularity, program analysis, type systems, specification, verification, security, real-time systems, etc.
- ☺ July 28      **11th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems (ICOOOLPS'2014)**. Topics include: implementation of fundamental OO and OO-like features (e.g. inheritance, parametric types, memory management, objects, prototypes), runtime systems (e.g. compilers, linkers, virtual machines, garbage collectors), optimizations (e.g. static or dynamic analyses, adaptive virtual machines), resource constraints (e.g. time for real-time systems, space or low-power for embedded systems) and relevant choices and tradeoffs (e.g. constant time vs. non-constant time mechanisms, separate compilation vs. global compilation, dynamic loading vs. global linking, dynamic checking vs. proof-carrying code...).
- ☺ July 28      **24th Doctoral Symposium**. Topics include: concurrency, real-time, embeddedness, distribution, language design, language constructs, static analysis, language implementation, virtual machines, methodology, model engineering, design languages, software evolution, formal methods, tools, programming environments, etc.

- ☺ July 29     **3rd Workshop on Combined Object-Oriented Modeling and Programming Languages (COOMPL'2014)**. Topics include: differences and similarities between modeling and programming; modeling constructs not supported by programming languages and vice versa; support for concurrent/distributed modeling and programming; tools for modeling and programming; implementation techniques; techniques for embedding domain specific languages in a combined language; new mechanisms to raise the level of abstraction; experience reports regarding pros/cons in using separate modeling and programming languages, modeling in a programming language, executable modeling languages, etc.
- August 04-07     **19th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'2014)**, Tianjin, China. Topics include: verification and validation, security of complex systems, model-driven development, reverse engineering and refactoring, design by contract, agile methods, safety-critical & fault-tolerant architectures, real-time and embedded systems, tools and tool integration, industrial case studies, etc.
- August 14-17     **Symposium on Dependable Software Engineering: Theories, Tools and Applications (SETTA'2014)**, Nanjing, China. Topics include: formal software engineering methods; formal aspects of engineering approaches to software and system quality; integration of formal methods into software engineering practice; formal methods for embedded, real-time, hybrid, and cyber-physical systems; formal aspects of security, safety, reliability, robustness, and fault-tolerance; model checking, theorem proving, and decision procedures; contract-based engineering of components, systems, and systems of systems; formal and engineering aspects of software evolution and maintenance; scalable approaches to formal system analysis and design; applications of formal methods and industrial experience reports; etc.
- August 20-22     **11th IEEE International Conference on Embedded Software and Systems (ICCESS'2014)**, Paris, France. Topics include: embedded real-time systems, distributed embedded computing, fault tolerant & trusted embedded systems, multicore systems, embedded real-time operating systems, cyber-physical systems, formal methods for embedded systems, middleware for embedded systems, compilation and debug techniques, IDE and software tools, robotics and control systems, automotive, medical and avionics systems, etc.
- August 20-22     **16th IEEE International Conference on High Performance Computing and Communications (HPCC'2014)**, Paris, France. Topics include: languages and compilers for high performance computing, parallel and distributed software technologies, parallel and distributed algorithms, embedded systems, tools and environments for software development, distributed systems and applications, high-performance scientific and engineering computing, reliability and fault-tolerance, trust, security, etc.
- ☺ Aug 24-27     **Communicating Process Architectures (CPA'2014)**, Oxford, UK. Theme: "36th WoTUG Conference on Concurrent and Parallel Systems". Topics include: all aspects of theory, design and implementation of concurrency in computer systems.
- ☺ Aug 25-29     **20th International European Conference on Parallel Computing (Euro-Par'2014)**, Porto, Portugal. Topics include: all aspects of parallel and distributed computing, such as support tools and environments, scheduling, high-performance compilers, distributed systems and algorithms, parallel and distributed programming, multicore and manycore programming, theory and algorithms for parallel computation, etc. Deadline for early registration: July 25, 2014.
- ☺ Aug 25     **7th International Workshop on Multi/many-Core Computing Systems (MuCoCoS'2014)**. Topics include: programming models, languages, libraries and compilation techniques; case studies highlighting performance portability and tuning; etc. Deadline for early registration: July 25, 2014.
- ☺ Aug 25     **1st International Workshop on Reproducibility in Parallel Computing (REPPAR'2010)**. Topics include: design, implementation, execution, and analysis of experiments in parallel computing in order to improve the reproducibility of results.
- ☺ Aug 25     **2nd Workshop on Runtime and Operating Systems for the Many-core Era (ROME'2010)**. Topics include: many-core aware runtime support for large-scale applications; dealing with legacy software on novel many-core architectures; experiences porting, running, or developing applications; traditional and new programming models for novel many-core hardware; bare-metal programming and system software; etc.

- ☺ Aug 26-28      **12th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA'2014)**, Milan, Italy. Topics include: parallel and distributed algorithms, and applications; high-performance scientific and engineering computing; middleware and tools; reliability, fault tolerance, and security; parallel/distributed system architectures; tools/environments for parallel/distributed software development; novel parallel programming paradigms; code generation and optimization; compilers for parallel computers; distributed systems and applications; scheduling and resource management; etc.
- August 27-29      **40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'2014)**, Verona, Italy. Topics include: information technology for software-intensive systems.
- August 29-31      **9th International Conference on Software Engineering and Applications (ICSOFT-EA'2014)**, Vienna, Austria. Topics include: software integration, software testing and maintenance, model-driven engineering, software quality, software and information security, formal methods, programming languages, parallel and high performance computing, software metrics, agile methodologies, risk management, quality assurance, certification, etc.
- September 01-03      **8th International Symposium on Theoretical Aspects of Software Engineering (TASE'2014)**, Changsha, China. Topics include: theoretical aspects of software engineering, such as specification and verification, program analysis, model-driven engineering, aspect and object orientation, embedded and real-time systems, component-based software engineering, software safety, security and reliability, reverse engineering and software maintenance, etc.
- September 01-05      **12th International Conference on Software Engineering and Formal Methods (SEFM'2014)**, Grenoble, France. Topics include: abstraction and refinement; programming languages, program analysis and type theory; formal methods for real-time, hybrid and embedded systems; formal methods for safety-critical, fault-tolerant and secure systems; software verification and validation; formal aspects of software evolution and maintenance; light-weight and scalable formal methods; tool integration; applications of formal methods, industrial case studies and technology transfer; education and formal methods; etc.
- September 07-10      **FedCSIS2014 - 7th Workshop on Computer Aspects of Numerical Algorithms (CANA'2014)**, Warsaw, Poland. Topics include: parallel numerical algorithms; libraries for numerical computations; languages, tools and environments for programming numerical algorithms; paradigms of programming numerical algorithms; etc.
- September 09-12      **11th International Conference on integrated Formal Methods (iFM'2014)**, Bertinoro, Italy. Topics include: the combination of (formal and semi-formal) methods for system development, regarding modeling and analysis, and covering all aspects from language design through verification and analysis techniques to tools and their integration into software engineering practice.
- ☺ Sep 09-12      **43rd Annual International Conference on Parallel Processing (ICPP'2014)**, Minneapolis, MN, USA. Topics include: all aspects of parallel and distributed computing, such as applications, architectures, compilers, programming models, etc.
- ☺ Sep 09      **International Workshop on Embedded Multicore Systems (EMS'2014)**. Topics include: programming models for embedded multicore systems; software for multicore, GPU, and embedded architectures; real-time system designs for embedded multicore environments; applications for automobile electronics of multicore designs; compiler for worst-case execution time analysis; formal method for embedded systems; etc.
- ☺ Sep 09      **5th International Workshop on Parallel Software Tools and Tool Infrastructures (PSTI'2014)**. Topics include: static and dynamic analysis tools; instrumentation, measurement, analysis, and modeling of applications; analysis and visualization tools for assisting programmers with parallel software design; etc.
- September 15-16      **7th International Conference on Software Language Engineering (SLE'2014)**, Vasteras, Sweden. Topics include: techniques for software language reuse, evolution and managing variation (syntactic/semantic) within language families; engineering domain-specific languages (for modeling, simulating, generation, description, checking); novel applications and/or empirical studies on any aspect of SLE (development, use, deployment, and maintenance of software languages); etc.
- September 15-17      **27th International Workshop on Languages and Compilers for Parallel Computing (LCPC'2014)**, Hillsboro, OR, USA. Topics include: parallel programming models, parallel programming languages,

compiling for parallelism and parallel compilers, formal analysis and verification of parallel programs, debugging tools for parallel programs, parallel applications, synchronization and concurrency control, software engineering for parallel programs, etc.

- September 15-19 **8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'2014)**, Turin, Italy. Topics include: qualitative methods, replication of empirical studies, empirical studies of software processes and products, industrial experience and case studies, evaluation and comparison of techniques and models, reports on the benefits / costs associated with using certain technologies, empirically-based decision making, quality measurement and assurance, software project experience and knowledge management, etc.
- September 17-20 **11th International Colloquium on Theoretical Aspects of Computing (ICTAC'2014)**, Bucharest, Romania. Topics include: principles and semantics of programming languages; relationship between software requirements, models and code; program static and dynamic analysis and verification; software specification, refinement, verification and testing; model checking and theorem proving; integration of theories, formal methods and tools for engineering computing systems; models of concurrency, security, and mobility; real-time, embedded, hybrid and cyber-physical systems; etc.
- September 22-25 **14th International Conference on Runtime Verification (RV'2014)**, Toronto, Canada. Topics include: monitoring and analysis of software and hardware system executions. Application areas include: safety/mission-critical systems, enterprise and systems software, autonomous and reactive control systems, health management and diagnosis systems, and system security and privacy.
- September 24-26 **14th Workshop on Automated Verification of Critical Systems (AVoCS'2014)**, Twente, the Netherlands. Topics include: model checking, specification and refinement, verification of software and hardware, specification and verification of fault tolerance and resilience, real-time systems, dependable systems, verified system development, industrial applications, etc. Deadline for submissions: August 7, 2014 (research ideas).
- Sep 28 – Oct 03 **30th IEEE International Conference on Software Maintenance and Evolution (ICSME'2014)**, Victoria, British Columbia, Canada. ICSME is the newly evolved ICSM.
- Sep 29 – Oct 01 **16th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'2014)**, Paderborn, Germany. Topics include: fault-tolerant and dependable systems, formal methods, safety, and security, cyberphysical systems, etc.
- © Oct 02-03 **CBSoft2014 - 18th Brazilian Symposium on Programming Languages (SBLP'2014)**, Maceió, Alagoas, Brazil. Topics include: the fundamental principles and innovations in the design and implementation of programming languages and systems; programming paradigms and styles, including object-oriented, real-time, multithreaded, parallel, and distributed programming; program analysis and verification, including type systems, static analysis and abstract interpretation; programming language design and implementation, including new programming models, programming language environments, compilation and interpretation techniques; etc.
- October 02-03 **14th International Conference on Quality Software (QSIC'2014)**, Dallas, Texas, USA. Topics include: software testing, software quality (review, inspection and walkthrough, reliability, safety and security, ...), static and dynamic analysis, validation and verification, economics of software quality, formal methods, component software and reuse, component-based systems, cyber-physical systems, distributed systems, embedded systems, safety critical systems, etc.
- October 06-09 **33rd International Symposium on Reliable Distributed Systems (SRDS'2014)**, Nara, Japan. Topics include: distributed objects and middleware systems, experimental or analytical evaluations of dependable distributed systems, formal methods and foundations for dependable distributed computing, high-assurance and safety-critical distributed system design and evaluation, secure and trusted distributed systems, etc.
- October 12-16 **9th International Conference on Software Engineering Advances (ICSEA'2014)**, Nice, France. Topics include: advances in fundamentals for software development; advanced mechanisms for software development; advanced design tools for developing software; software security, privacy, safeness; specialized software advanced applications; open source software; agile software techniques; software deployment and maintenance; software engineering techniques, metrics, and formalisms; software economics, adoption, and education; improving productivity in research on software engineering; etc.

- October 15-16     **6th International Workshop on Software Engineering for Resilient Systems (SERENE'2014)**, Budapest, Hungary. Topics include: requirements engineering & re-engineering for resilience; frameworks, patterns and software architectures for resilience; verification, validation and evaluation of resilience; empirical studies in the domain of resilient systems; etc.
- ♦ Oct 18-21     **ACM SIGAda Annual International Conference on High Integrity Language Technology (HILT'2014)**, Portland, Oregon, USA. Sponsored by ACM SIGAda, in cooperation with Ada-Europe and the Ada Resource Association. Co-located with SPLASH 2014. Deadline for submissions: July 5, 2014.
- ☺ October 20-24     **ACM Conference on Systems, Programming, Languages, and Applications: Software for Humanity (SPLASH'2014)**, Portland, Oregon, USA. Deadline for early registration: September 19, 2014.
- October 21-24     **14th International Conference on Formal Methods in Computer-Aided Design (FMCAD'2014)**, Lausanne, Switzerland. Co-located with MEMOCODE'2014 and DIFTS'2014. Topics include: theory and application of formal methods in computer-aided design and verification of computer systems and related topics; synthesis and compilation for computer system descriptions, modeling, specification, and implementation languages; model-based design; correct-by-construction methods; experience with the application of formal and semi-formal methods to industrial-scale designs; etc.
- November 03-06     **25th IEEE International Symposium on Software Reliability Engineering (ISSRE'2014)**, Naples, Italy. Topics include: reliability, availability, and safety of software systems; validation, verification, testing and dynamic analysis; software quality and productivity; software security; dependability, survivability, and resilience of software systems; open source software reliability engineering; supporting tools and automation; industry best practices; empirical studies; etc. Deadline for submissions: July 11, 2014 (tutorials), August 15, 2014 (workshop papers), August 25, 2014 (student papers), August 31, 2014 (fast abstracts).
- November 03-07     **16th International Conference on Formal Engineering Methods (ICFEM'2014)**, Luxembourg, Luxembourg. Topics include: abstraction and refinement; program analysis; software verification; formal methods for software safety, security, reliability and dependability; tool development, integration and experiments involving verified systems; formal methods used in certifying products under international standards; formal model-based development and code generation; etc.
- November 04-06     **14th International Conference on Software Process Improvement and Capability dEtermination (SPICE'2014)**, Vilnius, Lithuania. Topics include: process assessment, improvement and risk determination in areas of application such as automotive systems and software, aerospace systems and software, medical device systems and software, safety-related systems and software, financial institutions and banks, small and very small enterprises, etc. Deadline for early registration: September 1, 2014.
- November 16-21     **27th International Conference for High Performance Computing, Networking, Storage and Analysis (SC'2014)**, New Orleans, Louisiana, USA. Topics include: parallel algorithms, applications, distributed computing, performance, programming systems, system software, state-of-the-practice, etc. Deadline for submissions: July 31, 2014 (BOFs, Emerging Technologies, posters, showcases).
- November 16-22     **22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE'2014)**, Hong Kong, China. Topics include: architecture and design; components, services, and middleware; distributed, parallel, and concurrent software; embedded and real-time software; formal methods; model-driven software engineering; program analysis; reverse engineering; safety-critical systems; scientific computing; software engineering education; software evolution and maintenance; software reliability and quality; specification and verification; tools and development environments; etc.
- November 17-19     **12th Asian Symposium on Programming Languages and Systems (APLAS'2014)**, Singapore. Topics include: foundational and practical issues in programming languages and systems, such as semantics, design of languages and type systems, domain-specific languages, compilers, interpreters, abstract machines, program analysis, verification, model-checking, software security, concurrency and parallelism, tools and environments for programming and implementation, etc.
- November 27-28     **European Conference Software Engineering Education (ECSEE'2014)**, Seon Monastery, Germany. Topics include: new methods, techniques, best practices, and experiences in SE education; illustrative

examples to highlight SE topics in education; tools for SE education, both commercial and public domain; etc.

- December 01-04    **21st Asia-Pacific Software Engineering Conference (APSEC'2014)**, Jeju Island, Korea. Topics include: embedded real-time systems; formal methods; SE environments and tools; security, reliability, and privacy; software engineering methods; software maintenance and evolution; software process and standards; testing, verification, and validation; etc. Deadline for submissions: July 30, 2014 (industry track papers, postgraduate symposium papers, tutorials).
- December 08-12    **15th ACM/IFIP/USENIX International Middleware Conference (Middleware'2014)**, Bordeaux, France. Topics include: design, implementation, deployment, and evaluation of distributed system platforms and architectures for computing, storage, and communication environments, including reliability and fault-tolerance; scalability and performance; programming frameworks, parallel programming, and design methodologies for middleware; methodologies and tools for middleware design, implementation, verification, and evaluation; etc.
- December 10        **Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!**
- ☺ Dec 16-19        **20th IEEE International Conference on Parallel and Distributed Systems (ICPADS'2014)**, Hsinchu, Taiwan. Topics include: parallel and distributed applications and algorithms, middleware, multi-core and multithreaded architectures, scheduling, security and privacy, dependable and trustworthy computing and systems, real-time systems, cyber-physical systems, embedded systems, etc. Deadline for submissions: July 1, 2014 (papers).
- December 17-20    **21st IEEE International Conference on High Performance Computing (HiPC'2014)**, Goa, India. Topics include: parallel and distributed algorithms/systems, parallel languages and programming environments, hybrid parallel programming with GPUs and accelerators, scheduling, resilient/fault-tolerant algorithms and systems, scientific/engineering/commercial applications, compiler technologies for high-performance computing, software support, etc. Deadline for submissions: September 16, 2014 (student symposium submissions). Deadline for early registration: November 14, 2014.

## 2015

- 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.
- April 11-19        **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).
- ♦ 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) (requests pending).
- ☺ 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.
- December 10        **200th birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!**

**ACM SIGAda Annual International Conference**  
**High Integrity Language Technology HILT 2014**  
**Call for Technical Contributions**



**Developing and Certifying Critical Software**

October 18-21, 2014 — Portland, Oregon (USA)  
 Pre-conference tutorials: October 18-19  
 Conference: October 20-21; Co-located with *SPLASH 2014*



Sponsored by ACM SIGAda in cooperation with SIGBED, SIGCSE, SIGPLAN, SIGSOFT,  
 Ada-Europe and the Ada Resource Association

Contact: SIGAda.HILT2014 at acm.org [www.sigada.org/conf/hilt2014](http://www.sigada.org/conf/hilt2014)

**KEYNOTE SPEAKERS**



Tom Ball



Christine Anderson

*Thomas Ball* is a principle researcher in the area of Software Engineering at *Microsoft Research* where he manages the Software Reliability Research group. Tom was instrumental in the development of the SLAM model checker, and will be talking about “A Decade of Program Verification at Microsoft.” *Christine Anderson* was Manager of the Ada 9X Project, which completed its technical work 20 years ago to produce Ada 95; Christine is now Executive Director of *Spaceport America*, which is supporting commercial space flights by SpaceX and Virgin Galactic; she will be talking about her journey “From Ada9X to Spaceport America - Going Where No One Has Gone Before.” Other invited speakers to be announced soon.

**SUMMARY**

*High integrity* software must not only meet correctness and performance criteria but also satisfy stringent safety and/or security demands, typically entailing certification against a relevant standard. A significant factor affecting whether and how such requirements are met is the chosen language technology and its supporting tools: not just the programming language(s) but also languages for expressing specifications, program properties, domain models, and other attributes of the software or overall system. HILT 2014 will provide a forum for experts from academia/research, industry, and government to present the latest findings in designing, implementing, and using language technology for high integrity software. **We are soliciting technical papers, experience reports, and tutorial proposals on a broad range of relevant topics.**

**POSSIBLE TOPICS INCLUDE BUT ARE NOT LIMITED TO:**

- |   |  |
|---|--|
| <ul style="list-style-type: none"> <li>• New developments in formal methods</li> <li>• Multicore and high integrity systems</li> <li>• Object-Oriented Programming in high integrity systems</li> <li>• High-integrity languages (e.g., SPARK)</li> <li>• Use of high reliability profiles such as Ravenscar</li> <li>• Use of language subsets (e.g., MISRA C, MISRA C++)</li> <li>• Software safety standards (e.g., DO-178B and DO-178C)</li> <li>• Typed/Proof-Carrying Intermediate Languages</li> <li>• Contract-based programming (e.g., Ada 2012)</li> <li>• Specification languages (e.g., Z)</li> <li>• Annotation languages (e.g., JML)</li> </ul> | <ul style="list-style-type: none"> <li>• Model-based development for critical systems</li> <li>• Teaching high integrity development</li> <li>• Case studies of high integrity systems</li> <li>• Real-time networking/quality of service guarantees</li> <li>• Analysis, testing, and validation</li> <li>• Static and dynamic analysis of code</li> <li>• Information Assurance</li> <li>• Security and the Common Criteria / Common Evaluation Methodology</li> <li>• Architecture design languages (e.g., AADL)</li> <li>• Fault tolerance and recovery</li> </ul> |
|---|--|



## KINDS OF TECHNICAL CONTRIBUTIONS

**TECHNICAL ARTICLES** present significant results in research, practice, or education. Articles are typically 10-20 pages in length. These papers will be double-blind refereed and published in the Conference Proceedings and in *ACM Ada Letters*. The Proceedings will be entered into the widely consulted ACM Digital Library accessible online to university campuses, ACM's more than 110,000 members, and the wider software community.

**EXTENDED ABSTRACTS** discuss current work for which early submission of a full paper may be premature. If your abstract is accepted, a full paper is required and will appear in the proceedings. Extended abstracts will be double-blind refereed. In 5 pages or less, clearly state the work's contribution, its relationship with previous work (with bibliographic references), results to date, and future directions.

**EXPERIENCE REPORTS** present timely results and "lessons learned". Submit a 2-3 page description of the project and the key points of interest. Descriptions will be published in the final program or proceedings, but a paper will not be required.

**PANEL SESSIONS** gather groups of experts on particular topics. Panelists present their views and then exchange views with each other and the audience. Panel proposals should be 1-2 pages in length, identifying the topic, coordinator, and potential panelists.

**INDUSTRIAL PRESENTATIONS** Authors of industrial presentations are invited to submit a short overview (at least 2 page in length) of the proposed presentation and, if selected, a subsequent extended abstract for a 30-minute talk. The authors of accepted presentations will be invited to submit corresponding articles for *ACM Ada Letters*.

**WORKSHOPS** are focused sessions that allow knowledgeable professionals to explore issues, exchange views, and perhaps produce a report on a particular subject. Workshop proposals, up to 5 pages in length, will be selected based on their applicability to the conference and potential for attracting participants.

**TUTORIALS** can address a broad spectrum of topics relevant to the conference theme. Submissions will be evaluated based on applicability, suitability for presentation in tutorial format, and presenter's expertise. Tutorial proposals should include the expected level of experience of participants, an abstract or outline, the qualifications of the instructor(s), and the length of the tutorial (half day or full day).

**HOW TO SUBMIT:** Except for Tutorial proposals use <http://www.easychair.org/conferences/?conf=hilt2014>

<i>Submission</i>	<i>Deadline</i>	<i>Use Easy Chair Link Above</i>
Technical articles, extended abstracts, experience reports, panel session proposals, or workshop proposals	<b>June 7, 2014 now July 5!</b>	For more info contact: <b>Tucker Taft</b> , Program Chair taft@adacore.com
Industrial presentation proposals	<b>July 5, 2014</b> (overview) <b>Aug 6, 2014</b> (extended abstract)	
Send <b>Tutorial</b> proposals to	<b>June 7, 2014 now July 5!</b>	<b>John McCormick</b> , Tutorials Chair mccormick@cs.uni.edu

**At least one author is required to register and make a presentation at the conference.**

## FURTHER INFORMATION

**CONFERENCE GRANTS FOR EDUCATORS:** The ACM SIGAda Conference Grants program is designed to help educators introduce, strengthen, and expand the use of Ada and related technologies in school, college, and university curricula. The Conference welcomes a grant application from anyone whose goals meet this description. The benefits include full conference registration with proceedings and registration costs for 2 days of conference tutorials/workshops. Partial travel funding is also available from AdaCore to faculty and students from GNAT Academic Program member institutions, which can be combined with conference grants. For more details visit the conference web site or contact **Prof. Michael B. Feldman** (MFeldman@gwu.edu)

**OUTSTANDING STUDENT PAPER AWARD:** An award will be given to the student author(s) of the paper selected by the program committee as the outstanding student contribution to the conference.

**SPONSORS AND EXHIBITORS:** Please contact **Greg Gicca** (Gicca@Verocel.Com) to learn the benefits of becoming a sponsor and/or exhibitor at HILT 2014.

**IMPORTANT INFORMATION FOR NON-US SUBMITTERS:** International registrants should be particularly aware and careful about visa requirements, and should plan travel well in advance. Visit the conference website for detailed information pertaining to visas.

**ANY QUESTIONS?** Please send email to SIGAda.HILT2014@acm.org or Conference Chair Prof. Michael B. Feldman (MFeldman@gwu.edu) or Program Chair Tucker Taft (Taft@adacore.com).



Preliminary Call for Papers  
**20<sup>th</sup> 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  
 ETSIT-UPM  
[alonso@dit.upm.es](mailto:alonso@dit.upm.es)

#### Program co-Chairs

Juan A. de la Puente  
 ETSIT-UPM  
[jpuente@dit.upm.es](mailto:jpuente@dit.upm.es)

Tullio Vardanega  
 Università di Padova  
[tullio.vardanega@unipd.it](mailto:tullio.vardanega@unipd.it)

#### Tutorial Chair

Jorge Real  
 UPV  
[jorge@disca.upv.es](mailto:jorge@disca.upv.es)

#### Exhibition Chair

Santiago Uruña  
 GMV  
[suruena@gmv.com](mailto:suruena@gmv.com)

#### Industrial Chair

Jørgen Bundgaard  
 Rambøll Danmark A/S  
[jogb@ramboll.dk](mailto:jogb@ramboll.dk)  
 Ana Rodríguez  
 Silver Atena  
[ana.rodriguez@silver-atenas.es](mailto:ana.rodriguez@silver-atenas.es)

#### Publicity Chair

Dirk Craeynest  
 Ada-Belgium & KU Leuven  
[Dirk.Craeynest@cs.kuleuven.be](mailto:Dirk.Craeynest@cs.kuleuven.be)

#### Local Chair

Juan Zamorano  
 ETSIINF-UPM  
[jzamora@fi.upm.es](mailto:jzamora@fi.upm.es)



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

#### General Information

The 20<sup>th</sup> 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 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.



Francisco de Goya  
*La pradera de San Isidro* (1788)  
Museo del Prado, Madrid



FOR IMMEDIATE RELEASE

## **Ada 2012 Language Standard Published in Springer's LNCS and as Free eBook Further Widen the Availability of Latest Language Revision**

**BRUSSELS, BELGIUM, May 12, 2014.** Ada-Europe today announced the publication in extra formats of the 2012 version of the Ada programming language standard, after its formal approval by ISO/IEC JTC 1 in December 2012.

Since its standardization, the Ada 2012 standard has been available in HTML and Adobe Acrobat format (PDF), from the download sites [1] and [2]. More recently, the latest Ada language definition became also available as volume 8339 of Springer's Lecture Notes in Computer Science series [3], as a companion to the Ada 2012 Rationale, published by Springer as LNCS 8338.

Moreover, with a view to exploring new media platforms to further widen the availability of this important material, Ada-Europe has now produced a prototype eBook of the Ada 2012 Reference Manual, which can be downloaded from [1]. This eBook should be regarded as a draft concept, proposed for the scrutiny of the Ada community at large, for feedback on its perceived usefulness and suggestions for improvements. Returns on this subject should be addressed to Ada-Europe at [board@ada-europe.org](mailto:board@ada-europe.org).

[1] <http://www.ada-europe.org/resources/online>

[2] <http://www.adaic.org/ada-resources/standards/ada12>

[3] <http://www.springer.com/computer/swe/book/978-3-642-45418-9>

### **About Ada 2012**

Ada 2012 brings significant enhancements to Ada, most notably in the area of "contract-based programming." New features include the ability to specify preconditions and postconditions for subprograms, and invariants for private (encapsulated) types. These take the form of Boolean expressions that can be interpreted (under programmer control) as run-time conditions to be checked. The contract-based programming features fit in smoothly with Ada's Object-Oriented Programming model, and support the type substitutability guidance supplied in the Object-Oriented Technologies and Related Techniques Supplement (DO-332) to the new avionics software safety standard DO-178C / ED-12C.

Other new features in Ada 2012 include enhancements to the containers library, additional expressiveness through features such as conditional expressions and more powerful iterators, and support for multicore platforms (task affinities, and the extension of the Ravenscar profile - standardized in Ada 2005 as an efficient and predictable tasking subset for high-integrity real-time systems - to multiprocessor and multicore environments).

## About Ada-Europe

Ada-Europe is the international non-profit organization that promotes the knowledge and use of the Ada programming language in academia, research and industry in Europe. Its flagship event is the annual international conference on reliable software technologies, a high-quality technical and scientific event that has been successfully running in the current format since 1996.

Ada-Europe has member organizations all over the continent, in Belgium, Denmark, France, Germany, Spain, Sweden, and Switzerland, as well as individual members in many other countries. For information about Ada-Europe, its charter, activities and sponsors, please visit: [www.ada-europe.org](http://www.ada-europe.org). Ada-Europe is headquartered in Brussels, Belgium.

A PDF version of this press release is available at <http://www.ada-europe.org/>.

## Organization Contact

*Ada-Europe ivzw/aisbl*  
Tullio Vardanega, Ada-Europe President  
[president@ada-europe.org](mailto:president@ada-europe.org)

## Press Contact

*Ada-Europe ivzw/aisbl*  
Dirk Craeynest, Ada-Europe Vice-president  
c/o KU Leuven, Department of Computer Science  
[dirk.craeynest@cs.kuleuven.be](mailto:dirk.craeynest@cs.kuleuven.be)  
[vice-president@ada-europe.org](mailto:vice-president@ada-europe.org)



# Tools to get you there. Safely.

Ada and The GNAT Pro High-Integrity Family



[www.adacore.com](http://www.adacore.com)

**AdaCore**  
The GNAT Pro Company

# Feature Model Extraction from Documented UML Use Case Diagrams

**Mariem Mefteh, Nadia Bouassida**

Sfax University, Mir@cl Laboratory, Tunisia; email: [Mariem.mefteh.ch@gmail.com](mailto:Mariem.mefteh.ch@gmail.com),  
[Nadia.Bouassida@isimsf.rnu.tn](mailto:Nadia.Bouassida@isimsf.rnu.tn)

**Hanène Ben-Abdallah**

King Abdulaziz University, Jeddah, KSA; email: [HBenAbdallah@kau.edu.sa](mailto:HBenAbdallah@kau.edu.sa)

## Abstract

*The development of a feature model for a software product line requires a thorough domain analysis which needs a high expertise. Indeed, analysts must not only understand the requirements for one particular application, but they must also identify variable ways in which the requirements can be combined in all applications in the domain. Such an expertise being often hard to acquire, analysts need approaches that provide assistance based on any existing artefacts produced during the development of applications in the domain of the product line. In this paper, we present a fully automated approach that assists domain analysts in specifying the feature model of a software product line. Our approach exploits the use case diagrams of existing applications along with their textual documentation. Besides using the natural language documentation of the requirements, it has the merit of overcoming the possible incompleteness of such documentation.*

*Keywords: Feature model, SPL, textual scenario, UML use cases.*

## 1 Introduction

Maximizing the reuse of existing products has long been recognized as a means of cost reduction and quality improvement of software development. In fact, several reuse techniques have been proposed over the last decades including libraries, design patterns, components, frameworks, etc. Among the proposed techniques, we will focus in this paper on software product lines (SPLs).

According to Clements et al. [4], an SPL is “a set of software intensive systems sharing common, managed set of features that satisfy specific needs of a particular market segment and that are developed from a common set of core assets in a prescribed way”. Software development based on an SPL relies on assembling and configuring parts (called features) designed to be reused across the product line. While other reuse techniques support reuse by collecting a library of generic components with reuse potential, SPLs create reusable components needed in a predictive way to develop software within a particular product line. As such, this software development technique

promotes cost reduction, higher productivity, shorter time-to-market and higher quality products.

The main differences between the development of a conventional software and an SPL stem from the points of variation that this later must contain. The variation points represent the functional and behavioral differences among all software in the product line. They are the means of configuration of an SPL in order to derive a software from it, *i.e.*, to reuse it. Given their power to cover the whole domain of the product line, the identification of the variation points is the core of any development process of SPLs. These processes generally start from a set of assets pertinent to product variants in the product line and apply a set of decision rules to identify the common and variation points of the SPL, which are often modeled through a feature model [6]. Depending on the type of assets they use (source code or specification), the SPL development process adopts either a bottom-up or a top-down approach.

In a bottom-up approach, the feature model is derived from source code of product variants, whereas a top-down approach relies on a domain analysis to extract the feature model. Given the high expertise required in the second type of approaches, most proposed SPL development processes adopt a bottom-up approach. In addition, most existing bottom-up feature model extraction methods start from the source code of product variants (*cf.* [16], [20]). However, they have some difficulties in identifying all types of features and/or constraints related to the variation points. This limit stems in part from the low level of abstraction nature of the code. It can be overcome thanks to a thorough analysis of the domain. Instead of starting from scratch, which would require a high level of expertise, the domain analysis can be assisted by the requirements specification assets of existing product variants. Adopting this approach, the few proposed methods (*cf.* [19], [10], [1]) use textual documentation of product variants to derive feature models. Besides imposing a specific documentation template, similar to their bottom-up counterparts, these methods also have difficulties in identifying all the features and their variation constraints. Furthermore, they require an intense intervention of the designer.

In this paper, we present a fully automated top-down method that helps the experts in the domain analysis task in order to construct feature models. Our method relies on



documented UML use case diagrams as assets. Indeed, these are recently being explored as a way to model the domain of SPLs (*cf.* [8], [9]). Compared to existing methods, our approach is more applicable for two reasons: UML being a de facto standard software modelling language guarantees the availability of use case diagrams of product variants; and the textual documentation of the use cases describes interaction scenarios in a relatively standard format. Nevertheless, our method needed to face two main challenges: the possible incompleteness of use case diagrams and their textual documentation, and the natural language semantic ambiguities. To handle these challenges, our method uses a Formal Concept Analysis (FCA) method [3], the Semantic Model (SM) [17] and the trigger model [15] to extract relevant features and derive their hierarchies. In addition, the constraints among features are deduced by using semantic criteria and exploiting the “includes” relationships between pairs of use cases.

The remainder of this paper is organized as follows. Section 2 presents some fundamental concepts used in later sections. Section 3 overviews currently proposed approaches for features and feature model extraction from use case diagrams and/or textual descriptions. Section 4 presents our method and section 5 illustrates it through an example from the field of mobile media. Finally, section 6 summarizes the paper and presents an overview of our ongoing work.



## 2 Fundamentals

Before presenting our feature model extraction method, we overview the concepts of feature model, use case documentation, semantic model [17], and trigger model [15].


### 2.1 Feature models



Among the most popular formalisms used to model and reason about SPLs, feature models [6] are used to describe common and variation points. A feature model (FM) is a hierarchical graph where every node corresponds to a *feature* and each arc represents feature variability.

Informally, a feature is a characteristic of a system relevant for some stakeholder. It can be:

- “*mandatory*” (graphically represented as  ) which means that its presence is compulsory in every product configuration where its parent feature is present; or
- “*optional*” (graphically represented as  ) which means that its presence is not compulsory in every product.

Feature nodes in a feature model can be related by the following types of links:

- “*OR*” (graphically represented as  ): indicates that at least one of the child features has to be selected when deriving a specific product.

- “*XOR*” (graphically represented as  ): indicates that exactly one of the child features should be selected when deriving a specific product.
- “*Require*” (graphically represented as  $A \rightarrow B$ ): indicates that if the feature A is selected in a configuration, then the feature B must be selected too.
- “*AND*” (graphically represented as  ): indicates that two features must be selected together in the same configuration.
- “*Exclude*” (graphically represented as  $A \dashrightarrow B$ ): indicates that the features A and B should not to be part of the same configuration.

Any combination of features that does not violate the constraints of the feature model corresponds to a specific product [7].

### 2.2 Use case documentation

Developers resort to use cases (UC) as an intuitive means to express user requirements and understand the application domain. A use case instance can be expressed in terms of a scenario written in natural language, which explains in detail a specific way of using the system. In our work, we adopt the scenario template of the Cockburn’s use case taken from [5]. As explained in Table 1, the template is relatively simple yet rich.

**Table 1** Cockburn’s use case template

Use case name	< the name is the goal as a short active verb phrase >	
Goal in context	<a longer statement of the goal in context if needed >	
Scope	<what system is being considered black box under design >	
Level	<one of : Summary, Primary Task, Subfunction >	
Preconditions	<what we expect is already the state of the world >	
Success end condition	<the state of the world upon successful completion >	
Failed end condition	<the state of the world if goal abandoned >	
Primary actors	<a role name or description for the primary actor >	
Secondary actors	<other systems relied upon to accomplish use case >	
Trigger	<the action upon the system that starts the use case >	
Description	Step	Action
	1	<The steps of the scenario from trigger to goal delivery, and any clean up after >
	2	<... >
Extensions	Step	Branching Action
	1a	<condition causing branching > : <action or name of sub.use case >
Sub-Variations	Step	Branching Action
	1	<list of variations >



The use case detailed scenarios accompany the UML use case diagram as a documentation asset. In addition to the use case names, the UML diagram represents the preconditions and extensions of the use case scenarios through the "includes" and "extends" relations among the use case nodes. In the remainder of this paper, we call such UML use case diagram a documented use case diagram.

### 2.3 Semantic model

The goal of the semantic model (SM) [17] is to address the gap between how we think and how we shall resort to operational details to explain the same ideas in a natural language. It is based on a set of rules, called mapping rules, that represent a text written in any natural language in a formal and unique way, and that can be processed later to deduce important information like the corresponding source code [17].

For example, “*a doctor is a person*” (in English), “*un docteur est une personne*” (in French), “*ein Arzt ist eine Person*” (in German) have ultimately the same meaning. They correspond to a hierarchy concept relation that defines the two concepts “doctor” and “person”. The corresponding mapping rule is the following:

**(definition, hierarchy concept relation,  
(sub-concept, (doctor, (quantity, abstract))),  
(super-concept, (person, (quantity, abstract))))**

where: *definition* means that the concepts are cited for the first time; (*doctor, (quantity, abstract)*) and (*person, (quantity, abstract)*) represent, respectively, the name of the concept “doctor” with a semantic role *sub-concept*, and the name of the concept “person” with a semantic role *super-concept*; *hierarchy concept relation* corresponds to the type of the current mapping rule. In our case, the hierarchy concept relation is equivalent to the generalization in UML.

We will use the semantic model to detect, from the textual description, use cases that are missing in some of the use case diagrams of product variants. Detecting missing use cases can be vital during the validation of the user requirements.

### 2.4 Trigger model

The trigger model is a probabilistic model that was initially proposed by Lau et al. [15]. It is used to represent pairs of highly correlated words; that is, the occurrence of one of the two words in the history increases the prediction of the other. To determine the pairs of correlated words, the mutual information between words is calculated to measure the co-occurrence of words in a given context.

In our method, we use the trigger model for uses cases instead of words. Thus, the mutual information measure represents the co-occurrence of two use cases in the same use case diagram. It is calculated as follows:

$$MI(\omega_i, \omega_j) = \log \frac{P(\omega_i, \omega_j)}{P(\omega_i) \times P(\omega_j)} \quad (1)$$

where  $\omega_i$  and  $\omega_j$  are use cases;  $P(\omega_i, \omega_j)$  is the probability of finding the use cases  $\omega_i$  and  $\omega_j$  in the same use case diagram; and  $P(\omega_i)$  and  $P(\omega_j)$  are the probability of the use cases  $\omega_i$  and  $\omega_j$ , respectively.

The retained pairs of use cases are those whose mutual information exceeds a given threshold; they are considered as triggers.

## 3 Related work

Acher et al. [1] proposed a semi-automated approach to extract feature models from textual descriptions documenting a set of products. Their approach supposes that the documentation is organized in a tabular format where each row corresponds to a product and each column represents a product description. Similarly, Hartmann et al. [13] defined an approach that takes as input products documented as feature models or in a tabular format and produces a Supplier Independent Feature Model (SIFM). To be efficient, these two approaches require formal and complete descriptions in the predefined tabular format, which is not always the case. In addition, these approaches require user (domain analyst) intervention. Our aim is to overcome these two limits by automating the approach and extending, if necessary, the documentation to ensure the derivation of feature models from documented use case diagrams.

Weston et al. [19] proposed a tool that creates feature models from requirements specifications expressed in natural language, using clustering methods. In this approach, the variability information must be integrated manually into the resulting feature models and it cannot be synthesized automatically. Dumitru et al. [10] also used clustering methods to identify features from publicly available online specifications in any form. Their approach can discover domain-specific features and generate a probabilistic feature model as well as product specific feature recommendations. Our approach uses a particular kind of clustering technique through the Formal Concept Analysis (FCA) [3], a method of data analysis that describes relationships between a particular set of objects and a particular set of attributes. These latter constitute the input data of the FCA, represented in a tabular form, called cross table; the objects and the attributes correspond, respectively, to its rows and its columns. Formal concepts are particular clusters in cross-tables, defined by means of attribute sharing [3]. Their collection corresponds to a concept lattice.

Also based on textual documentation, Davril et al. [7] proposed an approach that generates automatically feature models from a set of informal and incomplete product descriptions. No one can deny that the work done in this approach was a challenge since it extracts complete feature models from informal and incomplete descriptions. However, this approach does not handle all types of variability constraints, namely the *Require*, *AND* and *Exclude* constraints.

Besides tabular and textual descriptions as assets in the derivation of feature models, use case diagrams have also been explored by a few researchers. Griss et al. [12] relied on the «includes» and «extends» relationships of the use case diagram in order to deduce the structure of the feature model. Likewise, Wang et al. [18] proposed a semi-automatic approach that converts a set of use cases into a Domain Feature Model. One main limit of these approaches is that they only use the relationships between the use cases to model the SPL. In particular, they do not exploit the semantics in the derivation of the constraints among the features. In addition to overcoming this limit, our method has two additional challenges to face: it fully automates the process; and it must be able to construct a complete feature model from a set of possibly incomplete use case diagrams and scenarios.

It is also worth noting that recent works propose to model an SPL with some extensions of UML use case diagrams along the feature models. For example, Gomaa [11] proposed to model features as use case packages in order to provide for the visualization of variants among use case specifications derivable from the feature model; Alférez et al. [2] proposed to specify the functional requirements with a use case model and the SPL features and variability information with a feature model. Hence, another advantage of our approach for feature model construction from documented use case diagrams is that it can produce a documented use case diagram for the SPL along with the feature model. The first model would be used for a better comprehension of the domain while the second model would be used for guiding the derivation of a particular product.

## 4 Our approach

Our approach consists of two main phases, namely pre-processing use cases and building the feature model (see Fig. 1).

### 4.1 Use cases pre-processing

This first phase gets four possible kinds of product variant assets: complete and documented use case diagrams, incomplete but documented use case diagrams use case diagrams with no scenarios, and/or just scenarios written in natural language. We suppose that the documentation is structured according to the use case template of Table 1. The goal of this phase is to refine and complete the use case diagrams based on their documentation. This phase is composed of four main steps.

#### 4.1.1 Use case diagram completion

This step aims to complete the product variants' use case diagrams. It examines the textual scenarios, if there are any, to identify use cases that are missing in some product variants' use case diagrams. It relies on the field “Goal in context” in the template documenting the use case scenarios. This field is in fact a reference to use cases included in product variant use case diagrams.

The identification of the missing use cases is done thanks to the semantic model. More specifically, for each use case scenario, first we apply the semantic model on its “Goal in context” field to obtain the corresponding mapping rule. The result of their treatment produces a list of use cases that must be included in some variant use case diagrams. All use cases in this list can be automatically added to the product variants' use case diagrams.

#### 4.1.2 Use case name unification

This step ensures that the use case collection has a unified vocabulary. To do so, it uses an unsupervised classification of the use cases based on the semantics of their names.

The classification of use case names starts with the extraction of the grammatical units from each use case name. This can be done automatically thanks to *wordnet::SenseRelate::Allword* [21]. Once the grammatical units are extracted, a similarity distance measure is computed to decide on the semantic class of each use case.

Several similarity distance measures have been proposed in the literature of information retrieval and classification. In our context, we use the widely used *cosine similarity*, also known as the *TF/IDF* (term frequency – inverse document frequency) similarity, in order to assign a weight to a term  $i$  in a document  $j$  as follows:

$$\omega_{ij} = tf_{i,j} \times idf_{i,j} = tf \times \log\left(\frac{m}{D(i)}\right) \quad (2)$$

Where:  $\omega_{ij}$  is the weight of the word  $i$  in the document  $j$  (corresponding to the use case name  $j$ );  $tf_{i,j}$  is the frequency of the word  $i$  in the document  $j$ ;  $m$  is the total number of documents in the collection; and  $D(i)$  is the number of documents where the word  $i$  occurs.

Thus, we have to dispose of queries and documents. In our case, a query will be made of units that compose a product variant's use case and a document will be made of the association of grammatical units that compose a product variant use case, added to their synonyms extracted from *WordNet* [22]. The computing of the terms weights should be completed with the calculation of a similarity measure which is the cosines, as follow:

$$Sim(d_i, q) = \cos(\vec{d}_i, \vec{q}) = \frac{\sum_{t_j \in T} \omega_{ij} \times \omega_{qj}}{\sqrt{\sum_{t_j \in T} \omega_{ij}^2 \times \sum_{t_j \in T} \omega_{qj}^2}} \in [0, 1] \quad (3)$$

Where  $d_i$  is the document  $i$ ;  $q$  is the query (corresponding the use case name candidate);  $(\vec{d}_i, \vec{q})$  is the angle between the vectors  $\vec{d}_i$  and  $\vec{q}$ ;  $\omega_{ij}$  is the weight of the term  $t_j$  in  $d_i$ ;  $\omega_{qj}$  is the weight of the term  $t_j$  in  $q$ ; and  $T$  is the set of terms contained in the documents. After performing this calculation, the documents (i.e. the use cases) that are similar to a query (i.e. having the highest value of

$\cos(\vec{d}_i, \vec{q})$ ) are grouped together and form one semantic class. Then, to unify the UC names having the same meaning, *i.e.* belonging to the same semantic class, we select a name for each class from the use case names list belonging to it.

4.1.3 Use case name refactoring

In this step, we introduce new names to the use cases of all use case diagrams in order to unify them according to the obtained names of classes.

4.1.4 Use case diagram refinement

The goal of this step is to find missing use cases which were not shown in the variant use case diagrams. As a consequence, we will detect them from incomplete corpus of use case diagrams thanks to the use of a probabilistic model, called the trigger model. Thus, we have to calculate the matrix of mutual information between each pair of use cases. Then, we deduce the couple of use cases that are strongly correlated from this matrix. This means that the apparition of one of them impacts the apparition of the other one in the same use case diagram. Thus, we deduce the missing use cases and their relationships with other ones in the use case diagrams.

At this stage, we obtain a set of completed and refined use case diagrams candidates.

4.2 Feature model construction

After refining and completing the collection of use case diagrams of product variants, our method proceeds with the construction of the feature model for the SPL. To do so, first, each use case is admitted as a feature. Secondly, the relationships and the constraints among these features are

identified through two main tasks: feature hierarchy extraction followed by feature constraints extraction.

4.2.1. Feature hierarchy extraction

This first step applies the Formal Concept Analysis (FCA) method [3] on the collection of features to obtain a lattice of features from which the initial mandatory features are automatically identified: they are at the top of the lattice. The remaining features are considered as optional. Among these latter, we can deduce other mandatory ones. In fact, we use the "hypernymy" and the "synonymy" semantic relationships between words to rearrange the cross-table: If Y is a hypernym or synonym of X, then we can migrate the column content of Y into those of X in the cross table because the apparition of Y implies that of X. The extraction of hypernyms can be done automatically through the WordNet [22] ontology. Once the cross table is rearranged, we apply the FCA another time on it and obtain the remaining mandatory features on top of the new lattice. The remaining features are all optional.

Along with the new mandatory features, we also identify the parent/child relationship: Y is a hypernym of X implies that X is the parent of Y. Up to this stage, we obtain a feature model with a complete hierarchy but missing its constraints.

4.2.2. Constraint extraction

This task uses the following four rules to identify relevant constraints among the identified features:

- R1 [Constraint "OR"]: Using the "Meronymy" relationship, we deduce the constraint "OR" between features. If we have Meronyms(A,B) and Meronyms(A,C), then there is an "OR" constraint between the features B and C.

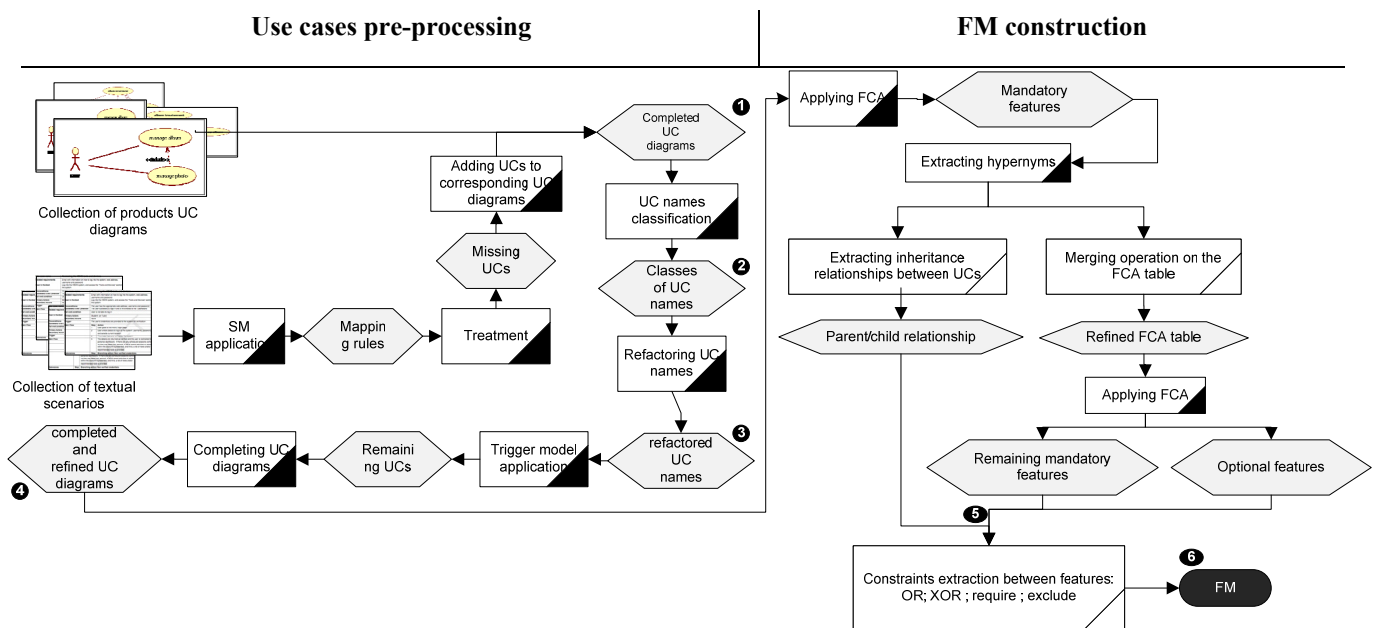


Figure 1 Automated two phase process: use cases pre-processing and FM construction

- *R2 [Constraint "XOR"]*: Using the "Synonymy" relationship, we deduce the constraint "XOR" between features if these latter have the same parent. Suppose that we have two features, *A* and *B*; if *A* and *B* are synonyms and they have the same parent, then we have an "XOR" relationship between them.
- *R3 [Constraint "Exclude"]*: Using the "Synonymy" relationship, we deduce the constraint "exclude" between features if these latter belong to two different parents.
- *R4 [Constraint "require"]*: Using the "include" relationship between two use cases (equivalent to features), we deduce the "require" relationship between them in the feature model.

After the extraction of relevant constraints among features thanks to the application of the previous rules, we obtain a complete feature model.

### 5 Case study

In this case study, we are interested in the construction of a FM from incomplete use case diagrams in the field of mobile media. We dispose initially of eight products that belong to the mobile media field. Due to space limitations, we will present only some of them (see Fig. 2).

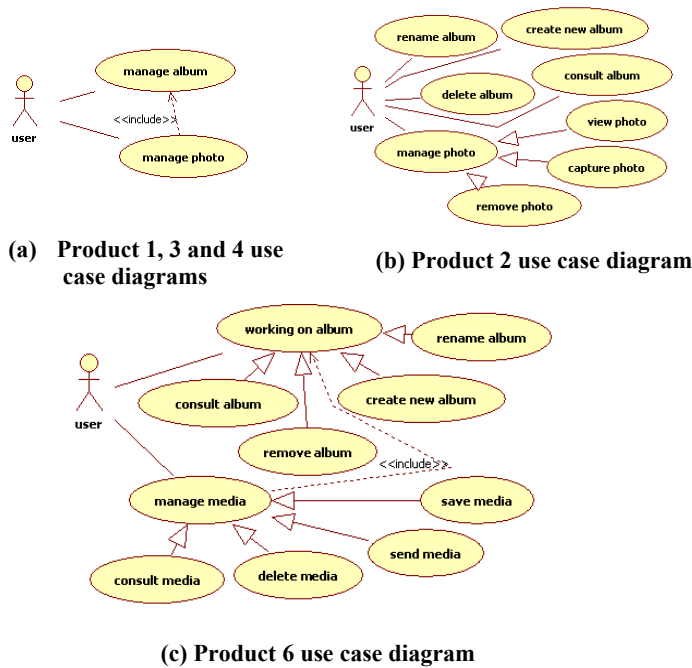


Figure 2 Presentation of the initial collection of use case diagrams belonging to 5 mobile media products

First of all, we complete the use case diagrams by the missing use cases, taking their textual descriptions as input. As presented in section 4.1.1, we rely on the field "Goal in context" to deduce them. For an example, let us consider the extract of the textual description of the use case "manage album" from the products 1, 3 and 4 which is shown in Table 2.

Table 2 An extract of the textual description of the use case "manage album" from the products 1, 3 and 4

UC name	manage album
Goal in context	The user can consult, create, rename or remove an album
....	....

Applying the semantic model on its goal, we obtain the following mapping rule:

(statement,  
 (action, (adjunctive, consult, create, rename, remove)),  
 (agent, (reference, explicit, user)),  
 (object,  
 (album, (quantity, abstract)) ) )

This rule follows this syntax (cf. [17] for more details):

(statement,  
 (class, predicate),  
 {(semantic role, argument)} )

Because the above rule contains a compression mechanism (expressed by the word "adjunctive") representing multiple actions, we deduce that the use case "manage album" is composed of the four sub-use cases, which are not explicitly drawn in the use case diagram of products 1, 3 and 4: consult album, create album, rename album and remove album. These missing use cases should be added to the use case diagram of product 1, 3 and 4 to complete it.

After applying the same step on the other use cases, we obtain all the missing use cases. By adding these use cases, we obtain a complete set of product variant's use case diagrams.

In the second step, we unify the use case names by grouping them into semantic classes. Initially, we started with the following use case names:

- |                              |                            |                     |                   |
|------------------------------|----------------------------|---------------------|-------------------|
| 1. manage album              | 10. move media             | 19. remove album    | 28. capture video |
| 2. consult album             | 11. save media             | 20. consult photo   | 29. send media    |
| 3. rename album              | 12. remove media           | 21. consult sound   | 30. play media    |
| 4. manage photo              | 13. consult media          | 22. save photo      | 31. play video    |
| 5. capture photo             | 14. delete video           | 23. capture picture | 32. view picture  |
| 6. view photo                | 15. remove video           | 24. record audio    | 33. play audio    |
| 7. remove photo              | 16. remove sound           | 25. save sound      | 34. manage media  |
| 8. delete album              | 17. delete sound           | 26. delete photo    | 35. delete media  |
| 9. album treatment           | 18. send sound             | 27. send photo      | 36. look up album |
| 37. working on album         | 40. create new album       |                     |                   |
| 38. processing album         | 41. create new photo album |                     |                   |
| 39. delete an existing media |                            |                     |                   |

The classification starts by extracting the grammatical units from each use case. For example, for the use case “manage album”, we obtain the grammatical units “manage” and “album”; for the use case “delete an existing media”, we obtain the grammatical units “delete”, “existing” and “media”. Afterwards, the TF/IDF similarity is computed using: 41 queries each of which is composed of the grammatical units extracted from one use case; and 41 documents each of which is composed of the grammatical units extracted from one use case along with their synonyms. As examples of queries, we have the following ones:

$q1 = \text{"manage album"}$        $q2 = \text{"manage photo"}$        $q3 = \text{"remove album"}$

And as example of documents, we have  $d1$  is the union of synonyms (“delete”) and synonyms (“album”), that is  $d1 = \{ \text{delete, cancel, remove, take, take away, withdraw, erase, take out, edit, blue-pencil, censor, album, record album, medium, book, volume} \}$

To apply the TF/IDF similarity measure, we first need to compute for each pair of document-query ( $d, q$ ) the weights of grammatical units of  $q$  in  $d$  according to equation (2). Then, name classes are automatically identified. For example, in the case of the pair ( $d1, q3$ ), we get:  $w_{delete, d1} = 0,84509804$  ;  $w_{remove, d1} = 0,62324929$  ;  $w_{delete, q3} = 0$  ;  $w_{remove, q3} = 1$  ; etc.

from which we deduce according to equation (3) that:

$$Sim(q3, d1) \approx cos(q3, d1) \approx 0,24600316.$$

After finishing the calculation of the similarity measures between all documents and queries, we obtain the set of classes grouping similar use case names shown in Table 3.

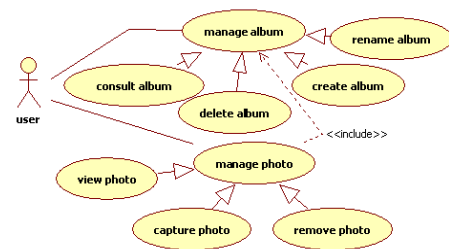
**Table 3 List of obtained semantic classes**

<b>C1:</b> play audio ; consult sound	<b>C2:</b> delete album ; remove album	<b>C3:</b> remove photo ; delete photo	<b>C4:</b> record audio ; save sound
<b>C5:</b> send photo	<b>C6:</b> manage media	<b>C7:</b> capture video	<b>C8:</b> rename album
<b>C9:</b> save media	<b>C10:</b> play media	<b>C11:</b> consult media	<b>C12:</b> play video
<b>C13:</b> create album ; create new photo album ; create new album	<b>C14:</b> remove media ; delete media ; delete an existing media	<b>C15:</b> view photo ; view picture ; consult photo	<b>C16:</b> manage photo
<b>C17:</b> capture photo ; capture picture ; save photo	<b>C18:</b> Manage album ; Work on album ; processing album ; album treatment	<b>C19:</b> remove video ; delete video	<b>C20:</b> Remove sound ; Delete sound
<b>C21:</b> send media	<b>C22:</b> send sound	<b>C23:</b> move media	<b>C24 :</b> consult album ; look up album

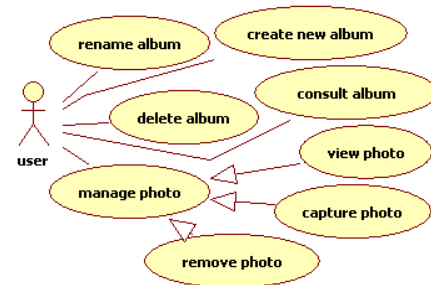
Based on this list, we attribute a name for each class reflecting its content. For example, we give the name “Manage album” to the class C18. When browsing our collection of use case diagrams, we rename every occurrence of the use cases “Work on album”, “album

treatment” and “processing album” with the name “Manage album”. Hence, our initial 41 use cases are reduced to 24 use cases.

The next task consists on refining use case diagrams by deducing probably remaining use cases thanks to the identification of triggers. This is done by the calculation of the MI matrix (see section 2.4). In our running example, we deduced the list of triggers containing (manage album; rename album), (manage album; consult album), (manage album; create album) and (manage album; Remove album). In each pair, the use cases have a high value of MI between each other. In other words, the occurrence of one use case of the trigger pair in a use case diagram increases the prediction of the other use case. In the running example, this leads us to deduce that the product 2 use case diagram, for example, is refined (see Fig. 3 instead of Fig. 4)



**Figure 3 Product 2 use case diagram after refinement**



**Figure 4 Product 2 use case diagram before refinement**

After applying this step on our collection of products, we obtain a set of completed and refined variant use case diagrams.

To build the feature model, we begin by applying the FCA on our collection (see Fig. 5). As we mentioned in section 4.2, every use case is considered as a feature. As a result, we get a lattice showing the mandatory ones on the top of the FCA lattice (see Fig. 6).

By examining the *hypernyms* of the remaining optional features, we can deduce other mandatory features. For instance, we have *hypernyms*(manage media, manage photo). Because “media” is the *hypernym* of “photo”, then “manage media” is the parent of “manage photo”; in other words, there is a hierarchical relationship between these two features in the FM. This means that each time we find “manage photo”, we implicitly have “manage media”. In addition, we notice in the FCA cross-table that the feature “manage photo” exists in the products 1, 2, 3 and 4, while “manage media” exists only in the products 5, 6, 7 and 8.

Consequently, we deduce that we can migrate the columns content corresponding to the feature "manage photo" to the corresponding one in "manage media" (see Fig. 7). This process is repeated for the other *hypernyms*.

After performing all the necessary migration operations, we obtain a new cross-table with a new lattice showing the appearance of new mandatory features (see Fig. 8). The remaining features are considered as optional.

Products	Manage album	Consult album	Rename album	Create album	Remove album	Manage photo	Capture photo	View photo	Send photo	Manage media	Save media	Remove media	Consult media	Move media	Send sound	Send media	Capture video	Play video	Play audio	Record audio	Remove photo	Play media	Remove video	Remove sound	
P1	X	X	X	X	X	X	X	X													X				
P2	X	X	X	X	X	X	X	X													X				
P3	X	X	X	X	X	X	X	X													X				
P4	X	X	X	X	X	X	X	X													X				
P5	X	X	X	X	X		X	X	X	X	X	X	X	X	X				X	X	X				X
P6	X	X	X	X	X		X	X		X	X	X	X			X			X	X	X				X
P7	X	X	X	X	X		X	X		X	X	X	X				X	X			X	X	X		X
P8	X	X	X	X	X		X	X		X	X	X	X				X	X	X	X	X	X	X	X	X

Figure 5 The FCA cross-table

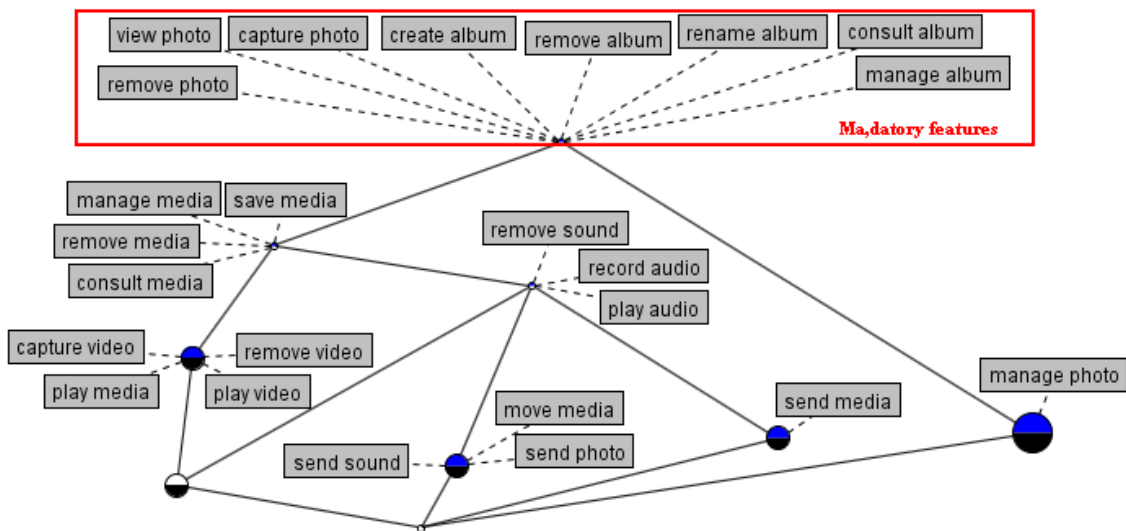


Figure 6 Lattice obtained after the first application of FCA on our collection

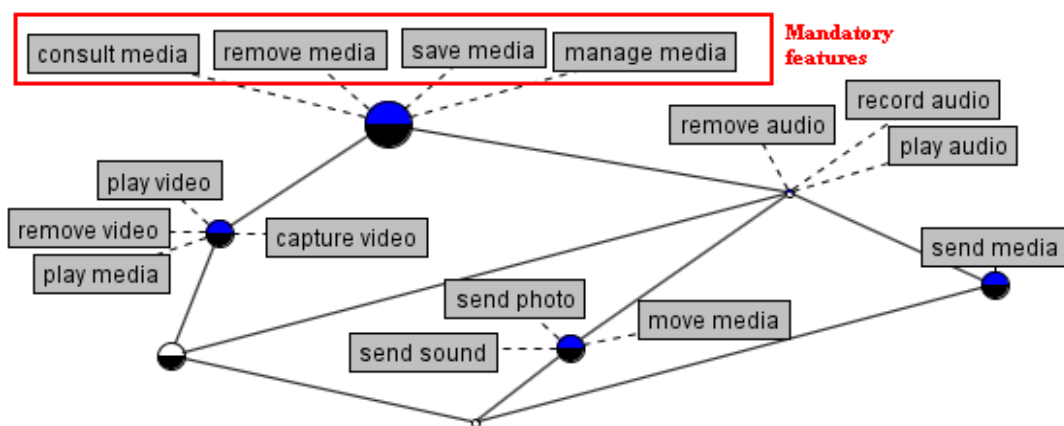


Figure 8 The lattice after the merging operations



Products	Manage album	Consult album	Rename album	Create album	Remove album	Manage photo	Capture photo	View photo	Send photo	Manage media	Save media
P1	X	X	X	X	X	X	X	X		X	
P2	X	X	X	X	X	X	X	X		X	
P3	X	X	X	X	X	X	X	X		X	
P4	X	X	X	X	X	X	X	X		X	
P5	X	X	X	X	X		X	X	X	X	X
P6	X	X	X	X	X		X	X		X	X
P7	X	X	X	X	X		X	X		X	X
P8	X	X	X	X	X		X	X		X	X

Figure 7 The migration operation from the feature “manage photo” to “manage media”

In the last step, we rely on the relations in the use case diagrams and the *hypernyms* list to extract parent/child relationships between features. We obtain a FM with a complete hierarchy but missing constraints. In order to deduce these latter, we have to apply the rules mentioned in section 4.2.2. In the following, we will illustrate some constraints extracted in our running example:

- We have the following meronyms: Meronyms(consult media, view photo); Meronyms(consult media, play audio) and Meronyms(consult media, play video). Applying the rule R1, we deduce that we have the constraint “OR” between the features “view photo”, “play audio” and “play video”.
- The use case "manage photo" includes "manage album" in the products 1, 2, 3 and 4. In addition, we note that the use case "manage media" includes the use case "manage album" in the products 5, 6, 7 and 8. Furthermore, since hypernyms(manage media, manage photo), we deduce that "manage media" includes "manage album" in all products. Applying the rule R4, we deduce that the feature "manage media" requires the feature "manage album".

After the extraction of relevant constraints between features, we obtain the complete FM shown in Fig. 9.

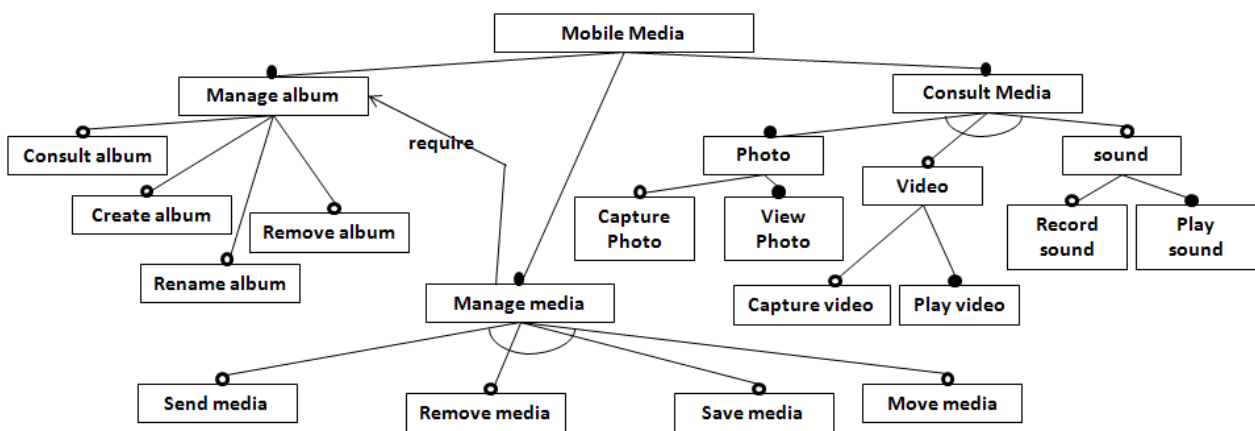


Figure 9 The resulting feature model

## 6 Conclusion

In this paper, we have presented a new top-down approach for extracting the FMs of SPLs from possibly incomplete descriptions. These latter are represented as UML use case diagrams with documented with textual scenarios. Our approach has the merit of using semantic information along with the structural information from the use case diagrams to produce automatically FMs. It uses various techniques in natural language processing to overcome the possible incompleteness of the textual description. In addition, it generates FMs with well-defined feature hierarchies and constraints.

We are in the process of developing the tool support for our approach in order to conduct an evaluation on a larger set of products. Another practical extension of the herein presented work is the derivation of source code for the features: We will explore the fact that textual scenarios include complete execution paths of the system in order to deduce a source code skeleton for each feature.

## References

- [1] M. Acher, M., A. Cleve, A., G. Perrouin, G., P. Heymas, C. Vanbeneden, P. Collet, P. Lahire (2012), *On extracting feature models from product descriptions*, VaMoS'12 Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, pp. 45-54.
- [2] M. Alférez, U. Kulesza, A. Moreira, J. Araújo, V. Amaral (2008), *Tracing between Features and Use Cases: A Model-Driven Approach*, Proceedings of the 2nd International Workshop on Variability Modelling of Software Intensive Systems (VAMOS), Essen, Germany.
- [3] R. Belohlavek (2008), *Introduction to formal concept analysis*, Olomouc.

- [4] P. Clements, L. Northrop (2001), *Software Product Lines: Practices and Patterns*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- [5] A. Cockburn (2001), *Writing Effective Use Cases*, Addison-Wesley, Reading, MA.
- [6] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, A. Wasowski (2012), Cool features and tough decisions: a comparison of variability modeling approaches, Proceedings of VaMoS'12, pages 173–182, New York, NY, USA.
- [7] J. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, P. Heymans (2013), *Feature Model Extraction from Large Collections of Informal Product Descriptions*, European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 290-300.
- [8] R. B. De Almeida (2010), *Modeling Software Product Line Variability in Use Case Scenarios \_ An Approach Based on Crosscutting Mechanisms*.
- [9] I. De Sousa Santos, R. M. de Castro Andrade, P. de Alcântara dos Santos Neto (2013), *A Use Case Textual Description for Context Aware SPL Based on a Controlled Experiment*, CAiSE Forum, volume 998 of CEUR Workshop Proceedings, CEUR-WS.org, pp.1-8.
- [10] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, M. Mirakhorli (2011), *On-demand feature recommendations derived from mining public product descriptions*, ICSE, ACM, pp. 181-190.
- [11] H. Gomma (2004), *Designing Software Product Lines with UML – From Use Cases to Pattern-Based Software Architectures*, Addison-Wesley.
- [12] M. L. Griss, J. Favaro, M. d'Alessandro (1998), *Integrating Feature Modeling with the RSEB*, Fifth International Conference on Software Reuse, Victoria, Canada.
- [13] H. Hartmann, T. Trew, A. Matsinger (2009), *Supplier independent feature modelling*, SPLC'09, IEEE, pp. 191-200.
- [14] C. W. Krueger (2006), *Introduction to the Emerging Practice of Software Product Line Development*, Methods & Tools, Fall issue, [http://www.methodsandtools.com/archive/archive.php?id=45\(2006\)](http://www.methodsandtools.com/archive/archive.php?id=45(2006)).
- [15] R. Lau, R. Rosenfeld, S. Roukos (1993), *Trigger-based language models: a maximum entropy approach*, Proc. ICASSP-93.
- [16] J. Maazoun, N. Bouassida, H. Ben-Abdallah, A. Seriai (2013), *Feature model extraction from product source codes based on the semantic aspect*, ICSOFT'2013, 8th International Conference on Software Paradigm Trends, Islande.
- [17] M. Mefteh, A. Ben Hamadou, R. Knöll (2012), *Ara\_Pegasus: A new framework for programming using the Arabic natural language*, International Conference on Computing and Information Technology, pp. 468-473.
- [18] B. Wang, W. Zhang, H. Zhao, Z. Jin, H. Mei (2009), *A Use Case Based Approach to Feature Models' Construction*, RE, IEEE Computer Society, pp. 121-130
- [19] N. Weston, R. Chitchyan, A. Rashid (2009), *A framework for constructing semantically composable feature models from natural language requirements*, SPLC'09, volume 446 of ICPS, ACM, pp. 211-220 .
- [20] T. Ziadi, L. Frias, M. A. A. da Silva, M. Ziane (2012), *Feature identification from the source code of product variants*, CSMR IEEE, pp. 417–422.
- [21] <http://maraca.d.umn.edu/allwords/allwords.html>
- [22] <http://poets.notredame.ac.jp/cgi-bin/wn>.



## Proceedings

# Workshop on Mixed Criticality for Industrial Systems

**Ada-Europe 2014**

**27 June 2014**

**Paris, France**

### Program

#### Keynote

"Correct-by-Construction Multiprocessor Programming: A Common Approach for Parallel and Mixed-Critical System Design"  
Albert Cohen, senior research scientist, PARKAS group, INRIA, France

#### Session 1

"PROXIMA: A Probabilistic Approach to the Timing Behaviour of Mixed-Criticality Systems"  
Robert Davis, Tullio Vardanega, Jan Andersson, Francis Vatrinet, Mark Pearce, Ian Broster, Mikel Azkarate-Askasua, Franck Wartel, Glenn Farral, Liliana Cucu-Grosjean, Mathieu Patte and Francisco Cazorla

"Toolset for Mixed-Criticality Partitioned Systems: Partitioning Algorithm and Extensibility Support"  
Alejandro Alonso and Emilio Salazar

"RTFM-lang Static Semantics for Systems with Mixed Criticality"  
Per Lindgren, David Pereira, Johan Eriksson, Marcus Lindner and Luís Miguel Pinho

#### Session 2

"Handling Criticality Mode Change in Time-Triggered Systems through Linear Programming"  
Mathieu Jan, Lilia Zaourar, Vincent Legout and Laurent Pautet

"Mixed Criticality over Switched Ethernet Networks"  
Olivier Cros, Frédéric Fauberteau, Laurent George and Xiaoting Li

"Mixed Criticality in Railway Systems: A Case Study on Signaling Application"  
Albert Cohen, Valentin Perrelle, Dumitru Potop-Butucaru, Elie Soubiran and Zhen Zhang

### Organization

**Chairs:** Laurent George, Luis Miguel Pinho

**Program Committee:** Hakan Aydin, Sanjoy Baruah, Guillem Bernat, Alfons Crespo, Liliana Cucu-Grosjean, Juan Antonio de la Puente, Sebastien Faucou, Joel Goossens, Leandro Indrusiak, Mathieu Jean, Claire Maiza, Moritz Neukirchner, Laurent Pautet, Zlatko Petrov, Eduardo Quinones, Yves Sorel, Benoit Triquet, Wang Yi

### Sponsored by



# PROXIMA: A Probabilistic Approach to the Timing Behaviour of Mixed-Criticality Systems

## **Robert I Davis**

Department of Computer Science, University of York, Deramore Lane, York, YO10 5GH, UK;  
email: rob.davis@york.ac.uk

## **Tullio Vardanega**

Department of Mathematics, University of Padova, Trieste 63, 35121 Padova, Italy;  
email: tullio.vardanega@math.unipd.it

## **Jan Andersson**

Aeroflex Gaisler AB, Kungsgatan, 12, 411 19 Göteborg, Sweden; email: jan@gaisler.com

## **Francis Vatrinet**

Sysgo SAS, Route de Sartrouville 54, 78230 Le Pecq, France; email: fva@sysgo.com

## **Mark Pearce, Ian Broster**

Rapita Systems Ltd., Atlas House, Osbaldwick Link Road, York YO10 3JB, UK;  
email: {mpearce, ianb}@rapitasystems.com

## **Mikel Azkarate-Askasua**

Ikerlan S.Coop. Paseo J. M. Arizmendiarieta 2, 20500 Mondragon, Spain;  
email: MAzkarateAskasua@ikerlan.es

## **Franck Wartel**

Airbus Operations SAS Route de Bayonne 316, 31060 Toulouse, France; email: franck.wartel@airbus.com

## **Liliana Cucu-Grosjean**

INRIA Paris-Rocquencourt, Domaine de Voluceau, BP 105 78153, Le Chesnay France;  
email: liliana.cucu@inria.fr

## **Mathieu Patte**

Astrium SAS, 31 rue des Cosmonautes – ZI du Palays 31402 Toulouse Cedex 4;  
email: mathieu.patte@astrium.eads.ne

## **Glenn Farrall**

Infineon Technologies UK Ltd, Infineon House, Great Western Court, Hunts Ground Road, Bristol, BS34 8HP, UK; email: glenn.farrall@infineon.com

## **Francisco J Cazorla**

Barcelona Supercomputing Center and IIIA-CSIC, c/Jordi Girona, 29, Edificio Nexus II. 08034 Barcelona, Spain;  
email: francisco.cazorla@bsc.es

## **Abstract**

*This position paper outlines the innovative probabilistic approach being taken by the EU Integrated Project PROXIMA to the analysis of the timing behaviour of mixed criticality real-time systems. PROXIMA supports multi-core and mixed criticality systems timing analysis via the use of probabilistic techniques and hardware/software architectures that reduce dependencies which affect timing. The approach is being applied to DO-178B/C and ISO26262.*

*Keywords: mixed-criticality systems; probabilistic real-time systems; WCET, software performance.*

## **1 Introduction**

EU industries developing Critical Real-Time Embedded Systems (CRTES), such as Aerospace, Space, Automotive, and Railways, face relentless demands for increased processor performance to support advanced new functionality. This demand is due to the ever-rising proportion of system value that is now delivered in software. For these industries, economic success depends on the ability to design, implement, qualify and certify

advanced real-time embedded systems within bounded effort and costs as well as pre-deployment assurance. Timing correctness as a means to guaranteed performance is one of the key dimensions of interest to qualification and certification for mission-, business- or safety-critical systems alike. Strong by-design evidence is therefore needed to build solid arguments of correctness that can satisfy certification bodies.

Over the next decade, CRTES industries in Europe will face a once-in-a-life-time disruptive challenge brought about by the transition to multicore processors and the architectural revolution that the advent of the manycore era brings. This step change in both processing capability and architecture (towards complex networked systems on a single chip), provides the opportunity to integrate multiple applications of mixed-criticality levels onto the same hardware platform. This has the advantages of reducing system size, weight and power consumption (SWaP), through a reduction in the number of devices, subsystems, and their cabling and connectors. Such integration has benefits in terms of reduced procurement costs, assembly costs, and improved reliability. However, the challenge also brings a severe threat relating to a key problem of CRTES. Unlike with conventional computing systems, developers of CRTES must provably demonstrate the correctness of the system in terms of both functional and timing/temporal behaviour. Current generation CRTES, based on relatively simple single-core processors, are already extremely difficult to analyse in terms of their temporal behaviour, resulting in incorrect operation that risks costing EU industry in high post-deployment costs (including “no-fault-found” and product recalls). The advent of multicore and manycore platforms exacerbates this problem, rendering timing analysis techniques unable to scale and ineffectual, with potentially dire consequences for the quality and reliability of future products. An innovative new approach is needed.

The PROXIMA approach is to adopt probabilistic analysis techniques to develop an efficient (tractable) and effective (tight) analysis of the temporal behaviour of complex mixed-criticality applications on novel and COTS (commercial-off-the-shelf) multicore and manycore platforms. Solid research results from the FP7 STREP PROARTIS ([www.proartis-project.eu](http://www.proartis-project.eu)) project [1] support this approach. The concept is based on using probabilistic analysis techniques [1, 2, 11, 12, 13] to derive tight bounds on the software timing behaviour of applications, reflecting requirements on failure rates commensurate with their criticality. PROXIMA defines architectural paradigms, usually based on the idea of randomizing the timing behaviour of hardware components, e.g. random replacement caches. These paradigms break the causal dependence in the timing behaviour of execution components at hardware and software level that can give rise to pathological cases, and reduces that risk of timing faults to quantifiably small levels. PROXIMA also supports COTS hardware components via the use of higher level (e.g., software-based) randomization paradigms [13] that

compensate for any probabilistic-analysis unfriendly features in them.

## 2 PROXIMA concepts

PROXIMA aims to enable the CRTES industry to successfully exploit the transition to multicore and manycore processor technology with a development approach that draws the most benefit and incurs the least disruption from it. Benefit will come from the ability to deploy more value-added, competitive-edge, heterogeneous and mixed-criticality functionality in more heavily integrated hardware platforms.

Containment of disruption will come from the ability to develop, analyse, build, and qualify CRTES incrementally. To meet that aim PROXIMA pursues an avenue of innovation relating to *composability in the time domain*, scalable across single-core, multicore and manycore processor architectures, without resorting to static partitioning and its intrinsic need for overprovisioning. Hence PROXIMA will solve a key challenge with mixed-criticality applications: the determination of trustworthy and tight bounds on the timing behaviour of applications. Thus low-criticality applications can be assured to not adversely affect higher-criticality ones while allowing for maximally efficient sharing of hardware and software resources among them, without the resource wastage inherent in fully deterministic approaches that use partitioning at every level.

The challenge is addressed by the use of probabilistic techniques, doing away with much of the need (and cost) of the detailed design knowledge required to causally model the timing behaviour of all system resources of interest. When the resource latency can be accurately captured with a probabilistic law and resource composition is designed to avoid causal dependence, the intrinsic complexity of novel multicore and manycore processor architectures naturally becomes treatable by probabilistic timing analysis.

### 2.1 High-performance mixed-criticality systems

PROXIMA is developing and exploiting innovative *probabilistic* analysis techniques and associated technology, to replace *deterministic approaches* originally designed for single-core processor systems that are rendered unsuitable or ineffectual with the advent of multicore and manycore architectures. This disruptive change makes current industrial practice inadequate for the development of the next-generation high-performance CRTES. Selective transformations are necessary for the development techniques and implementation technologies, which however can only be sustained if they minimise the cost of adoption. PROXIMA fosters that path of transformation.

The precursor PROARTIS project [1, 2, 11, 12, 13] has broken new ground in the domain of probabilistic timing analysis and paved the way to its application on single-core processors. In particular PROARTIS has shown that a wide range of probabilistic analysis techniques exist (including the theory of copulas, extreme value statistics, etc. [3]), that

can be applied to the timing analysis of real-time systems so long as certain assumptions apply, notably *statistical independence* (e.g. times are not dependent on the previous execution history) or some *definitional dependence* (times are solely defined by the software/hardware, e.g. constant time). It is important to note that these assumptions do not apply in most hardware/software architectures because the response time of resources (such as caches, pipelines) in modern processors is a (complex) function of the past history of use. Ironically, the fact that the behaviour of those resources is fully deterministic is of no benefit for the purposes of timing analysis. This is because the state space behind it is too vast to be precisely computed for single-core processors and is expected to be intractable for multicore and manycore systems.

The breakthrough strategy envisaged by PROXIMA is to introduce *architectural design principles* that result in temporal behaviour for which the hypothesis of either statistical independence or definitional dependence can be made to hold and therefore enables a meaningful application of probabilistic analysis. This fundamental property is achieved by moving away from deterministic behaviour to time randomised behaviour for jittery execution resources (e.g., cache, network-on-chip, memory allocation etc.) at both the hardware and software level without causing disturbance to the local and global functional behaviour affected by those resources.

## 2.1 CRTES criticality levels, probabilities, and failure rates

The use of probabilistic bounds in systems that require high assurance may seem counter-intuitive; however, the reality is that probabilistic modelling is a close match to the intrinsic nature of those systems. The mechanical parts of those systems (for example in aircraft) are designed with a failure rate in mind. This is so because effects such as radiation, mechanical stress and extreme temperatures induce a low, but non-zero and cumulative probability of failure for those parts and thus for the computing hardware itself. As a consequence, the system as a whole acquires a distinct probability of failure in a given time interval. This failure rate is measured in terms of the number of failures per hour (or billion hours).

By analogy, deviations in timing behaviour such as, for example the exceedance of given bounds in some execution time duration, may be considered as another type of failure that the system may experience. This reasoning should not be misrepresented as a shift in intent from designing software that meets its functional requirements to designing software that may fail in some well-defined way. Instead, it addresses the risk of execution time variability that originates from *outside* of the software itself, and stems from processor-level hardware resources whose innate jittery timing behaviour cannot be restrained by design other than at the cost of extreme overprovisioning.

The objective of probabilistic timing analysis is to provide WCET (worst case execution time) estimations and end-to-end worst-case response times (WCRT) that can be

determined to be “safe enough” with respect to application time constraints, so that they keep the overall failure rate of the application below the specific threshold of acceptability (e.g.  $10^{-9}$  per hour) for that application. Probabilistic and statistical approaches are a natural fit to mixed-criticality systems where applications at different criticality levels have different, domain-specific requirements in terms of acceptable timing failure rates, for example failure rates of  $10^{-7}$  per hour for low criticality and  $10^{-9}$  per hour for high criticality applications.

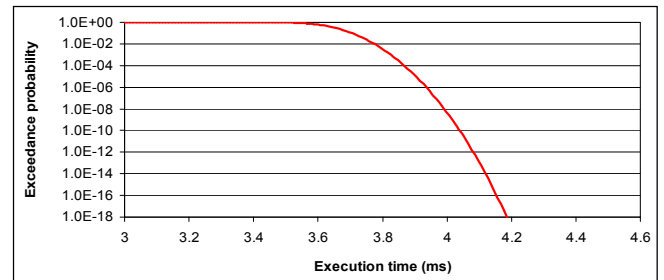


Figure 1 Probability of timing failure per hour

Probabilistic timing analysis provides a continuum of WCET bounds with associated probabilities of exceedance. By way of example, an application may have a probability of less than  $10^{-9}$ ,  $10^{-13}$  and  $10^{-18}$  of exceeding an execution time of 4.0ms, 4.1ms and 4.2ms, respectively, each time it executes— see Figure 1. Assuming, as a simple exemplar, that the execution time budget of the application is 4.1ms (for which its WCET has an exceedance probability of  $10^{-13}$  each time it executes), and that it executes at 50Hz (i.e. a 20ms period, or 180,000 times per hour) then its expected timing failure rate, due to budget overruns, is less than  $10^{-7}$  per hour, which may be acceptable for a low criticality application.

From this line of reasoning a close relation can be drawn with respect to criticality levels as defined, for example in avionics and automotive standards (DO-178B, ISO-26262) where a failure is defined as a deviation from a specified behaviour, the possible consequences of which determine its severity classification.

- In DO-178B, the Design Assurance level (DAL) is determined from the safety assessment process and hazard analysis by examining the effects of a failure condition in the system. The failure conditions are categorised by their effects on the aircraft, crew, and passengers, with comprehensive analysis methods used to establish the software level A-E: A Catastrophic, B Hazardous, C Major, D Minor and E no failure. Here, catastrophic failure must have a likelihood of occurring that is Extremely Improbably ( $<10^{-9}$ ) – as defined by FAA Advisory Circular AC-25-1309, whereas level B corresponds to Extremely Remote ( $<10^{-7}$ ).
- In ISO-26262, each Automotive Safety Integrity Level (ASIL) is associated with an observable incident rate. Hence applications of ASIL D must have an observable incidence rate lower than 1 every  $10^9$  hours, i.e.  $10^{-9}$  per hour. For ASIL C, B, and A the observable

incidence rate must be lower than  $10^{-8}$  per hour,  $10^{-8}$  per hour and  $10^{-7}$  per hour respectively.

The probabilistic approach of PROXIMA is a perfect match with those approaches. For applications with high criticality a probabilistic WCET (pWCET) estimate with a low probability of failure is chosen. (In the case of automotive this probability should be smaller than the incident rate in time defined by ISO-26262 multiplied by the number of times an application is executed per hour). In the case of DAL we take the pWCET estimate of each application based on its DAL. If a given application is DAL A, we ensure that the probability of that application having a failure in time is Extremely Improbably. Overall, the objective of probabilistic timing analysis is to provide WCET and end-to-end worst-case response time (WCRT) estimates which are ‘safe enough’ for the application, the meaning of which is determined by its criticality level.

As part of the PROXIMA project, a detailed analysis is planned examining the possible integration of project outcomes within certification standards and validation processes, with a certification authority acting as an external safety assessment reviewer.

### 3 Mixed criticality

Mixed-criticality CRTES bring a strong requirement to isolate the behaviour of applications in both the functional and time domains, otherwise the argument for integration is undermined because low criticality applications could impact those of high criticality to an unbounded extent, requiring all to be developed to the same rigorous, expensive and time consuming standard (appropriate for high criticality). To deal with this issue, and not increase verification and validation costs, industries from different domains have developed standardised software frameworks that provide elements of time isolation among software components on single-core processors (e.g. IMA in the avionics domain, and to some extent, AUTOSAR in the automotive domain). Both approaches support a hierarchical development process: the high level integration of the system should be straightforward from the composition of the timing behaviour of the software components. To do so, the system must support the *time composability* property: the worst-case timing behaviour of a component must not change (or only change predictably) when other components are integrated into the system. In multicore and manycore processors, this time composability property is not usually obtained because of the *dependences* on the execution time introduced by simultaneous access to shared resources. The execution time may vary greatly depending on the software components being run, i.e. depending on the system integration. Researchers have proposed to upper bound the maximum delay a software component can suffer due to interference when accessing shared resources such as buses [4, 5] or memory controllers [6]. For those resources where considering the maximum delay would remove the benefit of using them, e.g. cache, partitioning solutions have been considered [4].

Much of the recent research into mixed-criticality systems [10] owes its origins to the work of Vestal’s [7] which introduced varying degrees of WCET assurance, with larger WCET estimates obtained at higher levels of assurance (criticality level). This research shows that with a mixed-criticality system, simple reservation based policies such as time partitioning (discussed above), or allocation to processing cores based on criticality level can be inefficient; requiring significantly more processing resources than other appropriate scheduling approaches [8].

The alternative of using fixed priority scheduling (as used in automotive i.e. AUTOSAR) and assigning priorities based on criticality also results in severe resource under-utilisation [7, 9]. There is scope therefore for more sophisticated resource sharing policies and analyses to address the overprovision.

### 4 Time isolation and composition

With the advent of multicore and manycore processors, most complex CRTES are evolving into mixed-criticality systems. A key research question in mixed-criticality CRTES on these platforms is how to reconcile the conflicting requirements of partitioning for assurance and sharing for efficient resource usage [10].

PROXIMA addresses this question with respect to the twin requirements of time isolation and time composition. Asymmetric time isolation ensures that low criticality applications cannot adversely affect the timing behaviour of high criticality applications and hence do not need to be developed or verified to the same rigorous standards. Time composability ensures that the guaranteed timing behaviour of an application is not affected by the actual timing behaviour of other applications when the system is integrated. Together, time isolation and composability alleviate the effort and cost of system integration which is a major contributor to overall development costs, by permitting differential verification of software components added to a verified system. To date, timing isolation is normally accomplished via strict partitioning at all levels in the HW/SW stack; however, this comes at a high cost in terms of sizing for the worst case at every level, which while tolerable for single-core will prove unworkable with the transition to multicore and manycore.

The technology developed within the PROXIMA project attacks the root of the time composability problem by reducing, or even completely eliminating, the execution time dependencies resulting from sharing processor resources. As a result, the cost of acquiring the required knowledge to model the timing behaviour of the system can be reduced. In this way, software execution times are less dependent on previous and simultaneous execution of other software components and the system integration can be easily achieved.

The use of probabilistic approaches will recover the time composability property, avoiding the need to consider the maximum delay when accessing shared resources, or using time partitions. In the ideal case, if all the dependence on execution history is eliminated, each individual resource

will be time-composable, allowing software components to be replaced without requiring that the timing behaviour of other components is re-analysed.

PROXIMA technology also attacks the problem of overprovision intrinsic in simple partitioning and resource sharing approaches by providing hardware and software mechanisms and policies for resource sharing (between applications at the same and different criticality levels) that promote strong asymmetric isolation. This will minimise overprovision on two counts: firstly by enabling a structured abandonment of low criticality applications commensurate with their assurances and the rare need for high criticality applications to exceed a low assurance WCET budget defined for them. Secondly, by permitting effective resource reclamation when high criticality applications do not make use of their entire resource or WCET budget, permitting where feasible limited overrun capability for low criticality applications, improving their actual failure rates and hence perceived system quality.

## 5 Conclusions

In this short positional paper, we have outlined the innovative approach being taken by the PROXIMA project towards the analysis of future mixed-criticality real-time systems executing on multi- and many-core hardware platforms. PROXIMA has identified timing correctness as one of key dimensions of interest to qualification and certification of these mission-, business-, or safety-critical systems. The underlying concepts of PROXIMA involve the replacement of existing deterministic analysis techniques that are already reaching their limits on relatively simple single-core processors with more capable probabilistic analysis techniques. These techniques are supported by both hardware and software randomization that reduces the probability of pathological cases occurring to quantifiably low levels, that are significantly below the acceptable failure rates determined for the system.

## Acknowledgments

This work has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the PROXIMA Project ([www.proxima-project.eu](http://www.proxima-project.eu)), grant agreement number 611085.

## References

- [1] F. Cazorla, E. Quinones, T. Vardanega, L. Cucu, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim (2013), *PROARTIS: Probabilistically analysable real-time systems*, ACM Transactions on Embedded Computing Systems.
- [2] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzeti, E. Quinones, F. J. Cazorla (2012), *Measurement-based probabilistic timing analysis for multi-path programs*, In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS).
- [3] W. Feller (1996), *An introduction to Probability Theory and Its Applications*, Wiley.
- [4] M. Paolieri, E. Quinones, F.J. Cazorla, G. Bernat, M. Valero (2009), *Hardware Support for WCET Analysis of Multicore Systems*, In proceedings International Symposium on Computer Architecture.
- [5] J. Rosen, A. Andrei, P. Eles, Z. Peng (2007), *Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip*, In proceedings Real-Time Systems Symposium, pp 49-60.
- [6] B. Akesson, K. Goossens, M. Ringhofer (2007), *Predator: A predictable SDRAM memory controller*, CODESISSE.
- [7] S. Vestal (2007), *Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance*, In proceedings of the Real-Time Systems Symposium.
- [8] S.K. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie (2010), *Scheduling real-time mixed-criticality jobs*, In Proceedings of the 35th International Symposium on the Mathematical Foundations of Computer Science, volume 6281 of Lecture Notes in Computer Science, pp 90-101.
- [9] S.K. Baruah, A. Burns, R.I. Davis (2011), *Response Time Analysis for Mixed Criticality Systems*, In proceedings of the Real-Time Systems Symposium (RTSS), pp 34-43.
- [10] A. Burns and R. I. Davis (2013), *Mixed Criticality Systems – A Review*, Available from <http://www-users.cs.york.ac.uk/~burns/>
- [11] R.I. Davis, L. Santinelli, S. Altmeyer, C. Maiza, L. Cucu-Grosjean (2013), *Analysis of Probabilistic Cache Related Pre-emption Delays*, In proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), pp 168-179.
- [12] L. Kosmidis, E. Quiñones, J. Abella, T. Vardanega, F.J. Cazorla (2013), *Achieving Timing Composability with Measurement-Based Probabilistic Timing Analysis*, In proceedings International Symposium on Object / component / service-oriented Real-time distributed computing.
- [13] L. Kosmidis, C. Curtsinger, E. Quinones, J. Abella, E. Berger, F.J. Cazorla (2013), *Probabilistic timing analysis on conventional cache designs*, In Proceedings Design, Automation & Test in Europe pp.603-606.

# Toolset for Mixed-Criticality Partitioned Systems: Partitioning Algorithm and Extensibility Support

*Alejandro Alonso, Emilio Salazar*

*Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Spain; email: {aalonso, esalazar}@dit.upm.es*

## Abstract

*The development of mixed-criticality virtualized multi-core systems poses new challenges that are being subject of active research work. There is an additional complexity: it is now required to identify a set of partitions, and allocate applications to partitions. In this job, a number of issues have to be considered, such as the criticality level of the application, security and dependability requirements, operating system used by the application, time requirements granularity, specific hardware needs, etc. MultiPARTES [6] toolset relies on Model Driven Engineering (MDE) [12], which is a suitable approach in this setting. In this paper, it is described the support provided for automatic system partitioning generation and toolset extensibility.*

*Keywords: Real-time systems, Partitioned Systems, Mixed Criticality, Model Driven Engineering.*

## 1 Introduction

The increasing power of processing hardware makes it possible to integrate system functionality in just one processor, instead of using several ones. Although this has a number of advantages, it presents a major problem when developing complex embedded systems. It is common that these systems include applications with different criticality level. This type of systems is called mixed-criticality. This approach presents new challenges, as it is necessary to certify the whole system, even though there are parts that are no critical.

A suitable approach is based on system virtualization. A virtualization kernel or hypervisor allows the creation of partitions that are isolated. Applications with different criticality level are executed in different partitions in a safe way.

MultiPARTES is a FP7 project aimed at developing tools and solutions for building trusted embedded systems with mixed criticality components on multicore platforms. The approach is based on an innovative open-source multicore-platform virtualization layer based on the XtratuM hypervisor. A software development methodology and an associated toolset will be provided, in order to enable trusted real-time embedded systems to be built as partitioned applications, in a timely and cost-effective way.

XtratuM [10] [5] is based on para-virtualization, which means that a given operating system has to be adapted for being able to run on top of the hypervisor. This improves system performance and predictability, making it suitable for real-time systems. XtratuM has been designed for providing spatial and space isolation. Partitions scheduling is based on a cyclic policy, that it is statically generated, compliant with ARINC-653 [2]. It precisely states when each partition has to be executed. XtratuM also supports multi-core processors.

In this paper, some aspects of the MultiPARTES toolset [11] [1] are presented. Its main goal is to support the development of mixed-criticality multi-core partitioned systems. The toolset integrates a number of tools for supporting activities such as system modelling, system partitioning, validation, and system building.

## 2 Toolset requirements

The development of the toolset has been driven by the requirements specification in [7]. It was mainly defined by the consortium, which is composed by academia, research institutes, and industrial partners, from the automotive, railway, space, video surveillance, and wind power domains. This specification has been refined with the comments from experts in the project Advisory Board. The most relevant requirements are summarized below.

- *Development of mixed-criticality systems:* The toolset is aimed at supporting the development of mixed-criticality systems. This implies that the concept of criticality is central in the whole development process. The criticality level of each application has to be stated.
- *System model:* The toolset has to provide means for modeling the whole system, which includes the applications, platform, and any other information that the developer has to provide for performing the requested functionality.
- *Support for non-functional requirements:* Non-functional requirements are of great importance when dealing with embedded systems. Time, safety, and security, are of non-functional requirements that will be supported. The toolset has to provide means for specifying them, and validating their fulfillment.



- *Support for partitioned systems:* System partitioning is a fundamental activity on the target type of systems. However, there is little support in similar development tools. This toolset should generate system partitioning that has to be compliant with the system models and non-functional requirements.
- *Support for multi-core architectures:* The execution platform can be multi-core, as it is commonplace in current industrial systems. The toolset shall support modeling multi-core systems and assigning partitions to cores.
- *Validation and consistency:* The toolset performs a number of models transformations, and artefacts generation. An aim of this work is to ensure that these outcomes are valid with respect to the system requirements. These objectives are considered in the implementation of the transformers. In addition, the toolset allows the integration of validation tools for performing checks when required.
- *Support for legacy systems:* It is common in industry to have applications that have been developed in the past, perhaps with different methods and tools. The toolset will provide means for allowing the integration of this type of applications in the development flow.
- *Support system deployment:* Deployment is the last step required before running the system. When dealing with partitioned embedded systems, this implies the generation of a bootable software image that includes the hypervisor, the partitions, and their operating system and applications. The toolset supports system deployment by generating mechanisms for the automatic building of the system. System deployment also requires the configuration of XtratuM.

### 3 Toolset architecture

The main components of the toolset and data flows are depicted in figure 1. Their basic role is:

*System modelling:* It comprises the main input to the tool. It is composed by three models for describing the execution platforms, the applications, and the restrictions to be applied in the partitioning.

*Partitioning tool:* It is in charge of generating a system partitioning, that is described in the *deployment model*. It includes system partitions, the assignment of applications to partitions, and the characteristics of the partitions, including the operating system, processor time, memory, etc. The partitioning tool takes as input the system model. It has to consider information, such as the applications' criticality level, their required operating system and hardware devices, etc. Based on this information it generates a deployment model that meets the restrictions and some basic requirements.

*Validation:* Full correctness of a system partitioning may require complex checks that are difficult to integrate within a single tool. In addition, it is desirable for the toolset to be extended for supporting additional non-functional requirements. It is convenient to be able to use external validation tools that

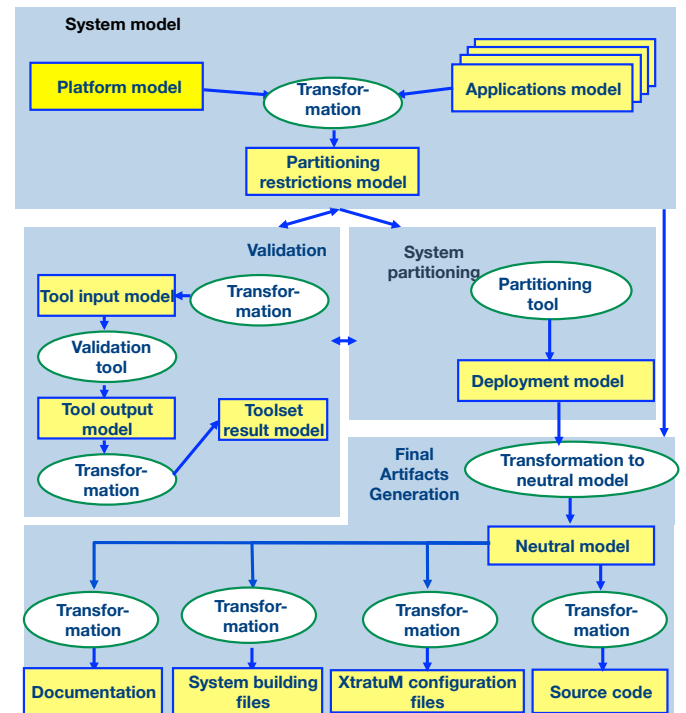


Figure 1: Overall architecture

check the correctness of the system configuration with respect to a given criteria.

*Generation of final artefacts:* when the system partitioning is correct a number of transformation tools generates a set of outcomes that are necessary for creating and building the final system:

- XtratuM configuration files
- System building files.
- Source code skeletons.

This toolset is currently under development. There is a working version that is able to handle simple models. Complexity is being added gradually. The toolset is being developed based on the Eclipse Modelling Tools. Model to model transformers are programmed in Query View Transformation Language (QVT). Model to text generators are based on Acceleo MTL. Metamodels are created using eCore.

### 4 Toolset extensibility

The MultiPARTES toolset has been designed for being easily extended and evolved. This has been a driver in the design of the architecture, shown in the previous section. There are a number of ways of enriching the current functionalities provided by the toolset, as adding support for additional non-functional requirements, validation tools, or tools for supporting system deployment.

The aim of this section is to describe the basic means for performing toolset extensions, as those mentioned. In fact, these facilities have been the basis for integrating in the core toolset the contributions developed by the partners in the project.

Toolset extension can be done at four main levels:



- System model level: to include in the model annotations for different non-functional properties, or other system aspects.
- Partitioning level: to convert annotations that have to do with partitioning into partitioning constraints.
- Validation level: to use external tools, a deployment model can be validated, according to the system model semantics, as stated by non-functional requirements annotations.
- Generation level: to generate code compliant with non-functional properties annotation or specific configuration parameters for XtratuM.

#### 4.1 Toolset extension at model level

System models can be annotated with information related with non-functional requirements. This is the case with the application model. Initially, all application models include information for partitioning and artefacts generation, such as criticality level or resources needed. Modelled applications rely on the class model in UML2 [8] for its description. The initial version of the toolset relies on the UML-MARTE [9] profile for describing time and resource requirements. In this case, it is possible to model real-time entities (tasks, protected objects) and real-time requirements and parameters. Application resource needs are derived from those of the individual entities.

Following this basis, additional annotations with respect to useful information for the developer can be added. If the information is associated to the application as a whole, then it can be enriched with annotations describing these new aspects. For example, it could be possible to mark an application as being of a specific type that requires specific handling by other tools.

In other cases, the annotations have to be made at the level of application components, such as classes, packages or threads. This case is more demanding. It requires the definition of a profile or metamodel, for defining the way and properties to be specified. Once again, other tools will have access to this information for performing their functions.

#### 4.2 Toolset extension at partitioning level

The partitioning tool is in charge of generating a system partitioning that is consistent with the policies for the different non-functional requirements. The proposed approach is to use the partitioning restrictions model as the basis for the integration of policies of different nature. For each non-functional property or developing aspect, a restrictions generator can be provided. It takes as inputs the platform and applications models, and generates a set of restrictions that ensures that the final system partitioning will meet the constraints in the policies. It is important to point out that the implementation language of the generator is not defined. Anyone can be used, provided that it generates valid restrictions, according to the provided meta-model.

Once all the restrictions derived from the different non-functional properties generators are available, the partitioning tool produces a system partitioning (deployment model) that is compliant with them, if one there exists.

#### 4.3 Toolset extension at validation level

The toolset allows the integration of additional validation tools. This can be required for supporting a new non-functional requirement, or performing a specific validation required in the development of a given system. The inputs to a new validation tool are system and deployment models. The outputs of the validation tool indicates to the partitioning tool whether the proposed deployment model is valid, and a set of new restrictions for driving its correct generation. It may be necessary to include new transformers for generating the validation tool input model or converting the corresponding output, for its integration in the toolset.

#### 4.4 Toolset extension at generation level

The aim is to allow the generation of additional artefacts useful for the development team. As mentioned above, currently the toolset generates XtratuM configuration files, system building files, and source code (Ada skeletons). However, there are additional artefacts that may be generated automatically, such as documentation or files for testing purposes.

Currently, the transformation components in the core toolset take as input the neutral model, for simplification purposes. However, new tools can access other models in the system, such as the system model or deployment model. A new transformation component can access this information for generating the desired artefacts. It is needed to ensure that the required information for this job is included in the models. This integration has to be performed at system model level. The toolset provides means for invoking them and their behaviour will be independent of other transformers, ensuring a straightforward integration.

### 5 System partitioning

The purpose of this section is to describe the general algorithm taken in the toolset for generating a feasible and automatic system partitioning. This component takes as input the system model: platform model, applications models, and partitioning restrictions model. This one is of particular interest, as it compiles restrictions that must be fulfilled by the resulting partitioning. They can be grouped in two types:

- Explicit: The developers and system integrator define this type of restrictions, which response to specific requirements. As instance, they can define specific hardware devices that must be used by an application, force specific allocations of application into partitions defined by the system integrator, etc.
- Implicit: They are automatically deduced from the system model. As mentioned in section 4, these restrictions are intended to ensure the fulfillment of non-functional requirements specified in the model. As instance, two applications with different criticality level cannot be in the same partition.

The output of this tool is a deployment model, which defines the system partitioning. It includes the description of the partitions. Each of them is characterized by the allocated applications, the used operating system, and required hardware resources.

The global approach for system partitioning relies on the divide and conquer principle. The complexity of this problem, and the requirements for extensibility are additional reasons for this approach. In consequence, the problem is broken down into four stages:

- Allocation of applications into partitions: The aim is to allocate all applications to partitions, trying to minimize its number, while fulfilling the restrictions.
- Allocation of partitions on processor cores. The result of this stage must meet the restrictions related with hardware devices.
- Cyclic plan scheduling design: As mentioned above XtratuM temporal isolation relies on a cyclic scheduling policy that is statically defined. The aim of this stage is to generate the cyclic plan, taking into account application allocation to cores, and applications CPU needs. These are defined in the applications model.
- Validation of the deployment model: Finally, it is possible to validate the resulting system partitioning with respect to general or non-functional requirements. This activity is performed by external tools that can be easily integrated in the toolset. For instance, a response time analysis tool is to be used [4]. Its aim is to ensure that time requirements are met by the proposed partitioning and scheduling plan.

The two initial stages are instances of the general allocation problem. It is NP-Hard, which means that there is no known algorithm that resolves it in polynomial time. This problem has been soundly researched for a long time. After making an analysis of some available options, the allocation problems on the partitioning algorithm in this toolset are based on the greedy algorithm of Iterated Register Coalescing (IRC) [3].

The IRC algorithm is based on colored graph theory. The allocation problem is modeled in a graph, where nodes represents the entities to allocate, colors are the allocation resources, and vertices represent restrictions. Originally, it was intended to help in the allocation of variables into hardware registers for code generation. There are a number of similarities, such as the existence of a number of restrictions that must be followed. This algorithm has been selected due to its good balance between the quality of the results and the implementation complexity.

The use of this approach for allocating applications on partitions required some adaptations. In the proposed solution, nodes represent applications, colors stand for partitions, and vertices are restrictions. The original IRC algorithm assumes a fixed number of resources (registers). However, in this allocation case, the number of resources (partitions) is not limited. Then, the proposed algorithm generates new colors when the allocation is not feasible or additional solutions are required.

The developed algorithm for the allocation of partitions to cores has also been adapted. The aim has been to prioritize solutions where the cores workload is balanced.

In addition, both algorithms have been improved with respect to the original IRC, in order to generate alternative solutions. A proposed system partitioning at this point may be invalid. This can be caused by not being able of generating a feasible cyclic plan or by failing in the validation stage. Then alternatives partitioning are generated, if it is feasible.

There is a working version of the two initial stages, which has been successfully tested with a number of system models. Work is ongoing for performing a more exhaustive and systematic test of these algorithms. There is a very simple version of the cyclic scheduling plan generator, that has been used in simple systems. A more advanced algorithm is currently under development.

## 6 Conclusions

This paper describes a toolset for supporting the development of mixed-criticality multi-core embedded systems. It relies on the XtratuM hypervisor that provides spatial and temporal isolation, as well as a number of additional features suitable for the development of this type of systems. The presented toolset has been designed according to a set of requirements produced by experts from academia and industry, with knowledge on a number of application domains.

Currently, the toolset provides most of the mentioned functionality, but for simple systems. Support for more complex systems is gradually being included. Future work includes the integration of improved support for time, safety and security, and improvements on the partitioning algorithm. The toolset is being validated in three different use cases: a wind-power turbine control system, the onboard software of the UPMSAT2 satellite, and a video surveillance system.

## Acknowledgment

The work in this paper is partially funded by FP7 STREP MultiPARTES project, no 287702 ([www.multipartes.eu](http://www.multipartes.eu)). The wish to thank the MultiPARTES consortium for its collaboration and help. The work in this paper has also been funded by the Spanish Ministerio de Educación, Cultura y Deporte, project HI-PARTES (High Integrity Partitioned embedded systems), TIN2011- 28567-C03-01 in the Plan Nacional de I+D+i.

## References

- [1] Alonso, A., Salazar, E., de Miguel, M.A. (2014), *A Toolset for the Development of Mixed-Criticality Partitioned Systems*, in 2nd Workshop on High-performance and Real-time Embedded Systems, Vienna, Austria
- [2] ARINC: Avionics Application Software Standard Interface ARINC Specification 653-1 (2003)
- [3] George, L., Appel, A.W. (1996), *Iterated register coalescing*, TOPLAS 18(3), 300–324.

- [4] González Harbour, M., Gutiérrez, J.J., Palencia, J.C., Drake, J.M. (2001), *MAST modeling and analysis suite for real time applications* in Proceedings of 13th Euromicro Conference on Real-Time Systems.
- [5] M. Masmano, I. Ripoll, A. Crespo, S. Peiro (2010), *Xtra-tuM for LEON3: an OpenSource Hypervisor for High-Integrity Systems*, Embedded Real Time Software and Systems (ERTS2 2010).
- [6] MultiPARTES: Multi-cores Partitioning for Trusted Embedded Systems, Available: [www.multipartes.eu](http://www.multipartes.eu)
- [7] MultiPARTES project, "Requirements Platform and Methodology Viewpoint", Deliverable D2.2, <http://www.multipartes.eu>.
- [8] OMG Unified Modeling Language (UML) (2011), <http://www.omg.org/spec/UML/2.4.1/>, version 2.4.1
- [9] OMG UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems (2011), <http://www.omg.org/spec/MARTE/>, version 1.1
- [10] S. Peiro, M. Masmano, I. Ripoll, and A. Crespo (2007), *PaRTiKle OS, a replacement of the core of RTLinux*, in Proc. of the Real-Time Linux Workshop.
- [11] E. Salazar, A. Alonso, M.A. de Miguel, J.A. de la Puente (2013), *A Model-Based Framework for Developing Real-Time Safety Ada Systems*, In H.B. Keller, et al (eds.), *Reliable Software Technologies — Ada-Europe*, LNCS 7896, pp. 126–141. Springer-Verlag.
- [12] Schmidt, Douglas C. (2006), *Guest editor's introduction: Model-driven engineering*, *Computer* 39.2 (2006): 0025-31.

# RTFM-lang Static Semantics for Systems with Mixed Criticality

*Per Lindgren, Johan Eriksson, Marcus Lindner*

*Luleå University of Technology, Sweden; email: {per.lindgren}@ltu.se*

*David Pereira, Luís Miguel Pinho*

*CISTER / INESC TEC, ISEP; email: {dmrpe,lmp}@isep.ipp.pt*

## Abstract

*In an embedded system, functions often operate under different requirements. In the extreme, a failing safety critical function may cause collateral damage (and hence consider to be a system failure) while non critical functions affect only the quality of service. Approaches by partitioning the system's functions into sandboxes require virtualization mechanisms by the underlying platform and thus prohibit deployment to the bulk of microcontroller based systems. In this paper we discuss an alternative approach based on static semantic analysis performed directly on the system specification expressed in the form of an object oriented (OO) model in the experimental language **RTFM-lang**. This would allow to (at compile time) to discriminate in between critical and non-critical functions, and assign these (by means of statically checkable typing rules) appropriate access rights. In particular, one can imagine dynamic memory allocations to be allowed only in non-critical functions, while on the other hand, direct interaction with the environment may be restricted to the critical parts. With respect to scheduling, a static task and resource configuration allows e.g. Stack Resource Policy (SRP) based approaches to be deployed. In this paper we discuss how this can be achieved in a mixed critical setting.*

## 1 Introduction

The number of embedded systems is rapidly increasing (according to the ARTEMIS SRA [1], we approach 40 billion devices in 2020). A vast major of these systems are based on small microcontrollers, providing limited CPU and memory resources. This requires tedious work by the programmer to household with the available resources where correctness is often validated by test based verification, which by nature is both time consuming and unsatisfactory with respect to safety critical functions. However, in many cases, only parts of the system's functions are safety critical (e.g., background functions such as monitoring, logging, etc. can be considered as non-critical, and allowed to fail momentarily as long as such failures do not (directly or indirectly) affect any critical function.

Moreover, systems may have a partly dynamic behaviour, e.g., if implementing a server, the number of connected clients may differ over time, and serving each client may require different amounts of memory during each session. Limiting the system to accommodate for the worst case behaviour in such a setting, either would imply too high resource requirements (leading to a costly, over dimensioned and potentially power hungry system) or impose overly pessimistic configurations (leading to bad performance/quality of service and poor utilisation of available resources). By allowing memory to be dynamically allocated, both utilisation and performance can be improved over the worst case setting. Thus, in such cases, a best effort approach is clearly preferable.

To this end, we have mainly two options: either we *prevent* operations (e.g., memory allocations) from failing, or we *allow* failure under certain conditions. Both options can be at least partially achieved in both a static, or a dynamic approach. Approaches (besides those of mere testing) such as model checking, or theorem proving are usually the ones used in a static setting, whereas runtime monitoring and runtime verification are able to indentify, correct, or simply minimize the effects that failures may exhibit upon execution. In general, static verification is hard, and often requires expert knowledge, thus is yet to reach into the mainstream of embedded system design. The same goes for run-time verification, witch in addition of adds both complexity and overhead to the implementation.

That gives motivation to investigate alternative approaches. In this paper, we claim that based on an appropriate language design, sought properties can be proven directly on the system model by means of (compile-time) analysis of its static semantics discriminating in between *critical* and *non-critical* functions. The compositional properties of **RTFM-lang** allows us to ensure, at compile time, that failures will be contained within the scope of non-critical functions. Such an approach could largely simplify the task of static verification.

Additionally we discuss, how a static task and resource configuration can be extracted from a mixed critical model in **RTFM-lang**. The task and resource configuration can be utilised for Stack Resource Policy based analysis and runtime scheduling. We show that, under fixed priority (single core) scheduling by the **RTFM**-kernel, response times for

critical functions are robust to overload of non-critical functions.

The presented approach is a work in progress. A subset of the **RTFM-lang** has been captured in the  $\mathbb{K}$  framework [2] with the aim of providing a formal semantics for both static (syntax and type system) as well as dynamic (run-time) behaviour of **RTFM-lang** programs. The  $\mathbb{K}$  framework provides a constructive approach to prototyping and development of programming languages. From the *syntax* a parser is directly obtained (giving immediate feedback on the specified grammar allowing for rapid prototyping). The derived abstract syntax trees can be processed by means of formally grounded term-rewriting rules [3]. These can be devised for example to reflect static semantics (well-formedness/type checking), dynamic semantics (run-time behaviour), as well as transformations into other models and languages. Each rule is matched and applied as an atomic operation, while the set of rules are applied non-deterministically. Using the  $\mathbb{K}$  framework one can model concurrent/parallel semantics, observe potential traces of non-deterministic behaviour, and even rely on the power of model checkers and/or automatic theorem provers to check properties of interest.

## 2 RTFM-lang Static Semantics

The overall of goals of developing **RTFM-lang** can be summarised as follows:

- An OO language with asynchronous and synchronous communication;
- Static object and communication structure related to critical functions;
- Dynamic object and communication structure related to non-critical functions;
- Failure free critical functions;
- Failure contention within non-critical functions.

### 2.1 Static Semantics Approach

In this paper we sketch a syntactic approach under the  $\mathbb{K}$  framework. The generated parser is able to discriminate between critical and non-critical functionality and syntactically enforce some of the above mentioned properties. However, one can do no so much at the level of parsing. To enforce well-formedness (type checking etc.), compilation techniques are of course mandatory. The syntactic approach, should be seen just as a proof of concept: for the future, we foresee to step away from the syntactic approach in order to provide a much more clean and succinct grammar for **RTFM-lang**. For instance, in the grammar (Figure 1) some duplications are introduced to enforce syntactic discrimination.

The sketched grammar merely captures object creation and communication aspects of the language, whereas actual computations, OO features, etc. are left out. The grammar should however be sufficient for the following discussion.

### 2.2 Static Object and Communication Structure

A distinct feature of the **RTFM-lang** is that a static (initial) set of objects can be declaratively defined, and spanned at compile time. Functions that are implemented solely in terms of operations on/within this static set of objects can be considered *safe* (e.g., with respect to memory accesses). The proposed language allows the programmer to explicitly state which methods should be considered *critical*, whose *safe* nature must be enforced/proved at compile time. This amounts to (transitively) require *critical* functions to rely solely on (other) critical functions. Since we assume that critical functions are safe, the overall safety is given by induction on the structure of the program and compositionality (this relates to well known assumption/guarantee approaches used in contract based program analysis [4]). Formalisation of contracts, and their verification is a work in progress.

#### 2.2.1 Example 1

The class definition specifies a provided and required interface. The provided interface is the set of public methods, while the required interface defines the type signature of instantiation parameters and callbacks.

In Figure 2 we depict the definition of classes **A** and **B**.

The declaration of `class A` requires a callback method `scb` as argument (specified in the '`<`' and '`>`' syntactic scope). The signature indicates that the callback is:

- **critical**, hence it can be used internally within critical functions;
- **sync**, meaning it is synchronous and returns a value);
- it takes an `int` argument and returns an `int` value.

`class A` provides a (implicitly public) method `m` with the same signature as the callback method `scb`. The method will return with the value of the synchronously executed `scb(j)`, where `j` was given on invocation. `class A` is also well-formed since we can (syntactically) deduce that all references from critical methods are to other critical methods (and the contract holds).

Looking at the definition of `class B` we find that it creates two instances `A<a2.m> a1` and `A<a1.m> a2` which are mutually dependent. Given that `class B` is instantiated statically, `a1` and `a2` are also static and their interdependency can be resolved at compile time. The use of `a1.m(1)` within `async trig()` is safe, since the contract requirement from a non-critical method is weaker (that is, non-critical methods may safely invoke methods from critical declarations). Moreover, this example shows that we can statically determine both the object structure (set of instances) as well as the communication topology, allowing us in this way to statically verify contracts for safe composition. The syntactic approach however, is insufficient to verify the dereferencing of objects.

#### 2.2.2 Example 2

In Figure 3 a new definition of class **A** presented, where objects are created dynamically (in a mixed critical setting). The definition is well-formed as (potentially unsafe) memory allocations are allowed in the non-critical context of `createA()`.

```

SYNTAX Prog ::= ClassDefs
SYNTAX ClassDefs ::= List{ClassDef, ""}
SYNTAX ClassDef ::= class Id < ClassArgs > {ClassVarDecls MethodDefs}
SYNTAX ClassArgs ::= List{ClassArg, ""}
SYNTAX ClassArg ::= Type Id
                    | MType Id(MethodSig)
SYNTAX MethodSig ::= List{PType, ""}
SYNTAX ClassVarDecls ::= List{ClassVarDecl, ""}
SYNTAX ClassVarDecl ::= PVarDecl
                    | OVarDecl
SYNTAX PVarDecl ::= PType Id ;
SYNTAX OVarDecl ::= Id < Exps > Id ;
SYNTAX MethodDefs ::= List{MethodDef, ""}
SYNTAX MethodDef ::= CMTDef
                    | MDef
SYNTAX CMTDef ::= critical MSyncAsync Id(CMArgs)CMBody
SYNTAX CMArgs ::= List{CMArg, ""}
SYNTAX CMArg ::= Type Id
SYNTAX MDef ::= MSyncAsync Id(MArgs)MBody
SYNTAX MArgs ::= List{MArg, ""}
SYNTAX MArg ::= PType Id
SYNTAX MType ::= MSyncAsync
                | critical MSyncAsync
SYNTAX MSyncAsync ::= sync Type
                    | async
                    | async Int
SYNTAX Type ::= PType
                | Id
SYNTAX PType ::= int
                | bool
                | char
SYNTAX MVarDecls ::= List{MVarDecl, ""}
SYNTAX MVarDecl ::= PVarDecl
                | OVarDecl
SYNTAX Stmt ::= MBody
                | Exp ;
                | send Int Exp ;
SYNTAX Stmts ::= Stmt
                | Stmts Stmt
                | return Exp ;
SYNTAX MBody ::= {}
                | {MVarDecls Stmts}
SYNTAX Exp ::= Int
                | Id
                | this
                | Exp . Id
                | Exp(Exps) [strict(2)]
                | Exp = Exp [strict(2)]
SYNTAX Exps ::= List{Exp, ""} [strict]
SYNTAX CMVarDecls ::= List{CMVarDecl, ""}
SYNTAX CMVarDecl ::= PVarDecl
SYNTAX CStmt ::= CMTDef
                | CExp ;
                | send Int CExp ;
SYNTAX CStmts ::= CStmt
                | CStmts CStmt
                | return CExp ;
SYNTAX CMBody ::= {}
                | {CMVarDecls CStmts}
SYNTAX CExp ::= Int
                | Id
                | this
                | CExp . Id
                | CExp(CExps) [strict(2)]
                | CExp = CExp [strict(2)]
SYNTAX CExps ::= List{CExp, ""} [strict]

```

Figure 1: Sketched grammar for a subset of RTFM-lang in  $\mathbb{K}$

```

1 class A <critical sync int scb(int)> {
2   critical sync int m (int j) {
3     return scb(j);
4   }
5 }
6
7 class B <> {
8   A<a2.m> a1;
9   A<a1.m> a2;
10
11  async trig() {
12    a1.m(1);
13  }
14 }

```

Figure 2: Definition of classes A and B

```

1 class A <critical sync int scb(int)> {
2   sync int createAs() {
3     A<a2.m> a1;
4     A<a1.m> a2;
5
6     return a1.m(29);
7   }
8
9   critical sync int m(int j) {
10    return scb(j);
11  }
12 }

```

Figure 3: Defintion of class A with mixed critical behavior.

### 2.2.3 Example 3

Figure 4 depicts an ill-formed definition of class A. It contain two errors, however, the grammar specified in the  $\mathbb{K}$  framework is only able to report one of the those. The attempt to create objects dynamically within the `critical` method `createAs` can be directly spotted. This is detected by the grammar through the rule

$$\text{SYNTAX } CMVarDecl ::= PVarDecl$$

which allows only primitive types to be locally allocated. The other error is the attempt to synchronously invoke the callback `scb(j)` (within the context of the critical method `m`) while it has been declared as being non-critical in the last argument. This goes beyond purely syntactic checking and requires type lookup in the scope of declared identifiers. This is by no means anything strange, and is target for ongoing work.

## 2.3 Safety and Robustness

In general, safety is about ensuring that nothing bad may occur, while robustness is a matter of gracefully dealing with the unexpected. To this end, memory management is a major obstacle. Whereas manually managing memory is known to be tedious and error prone, dynamic memory management has still gained limited use in the context of safety critical systems. Main challenges and obstacles to automatic methods are to bind overhead and prove memory sufficiency. By the use of virtualisation techniques, separation is possible (sand boxing critical functions), and by the use of hypervisor techniques resources (e.g., CPU) can be budgeted, such

```

1  class A <sync int scb(int)> {
2      critical sync int createAs() {
3          A<a2.m> a1;
4          A<a1.m> a2;
5
6          return a1.m(29);
7      }
8
9      critical sync int m(int j) {
10         return scb(j);
11     }
12 }

```

Figure 4: Ill-formed definition of class A.

that critical parts are ensured to operate correctly. In such a setting, critical partitions are typically static (or limited to a set of modes of operation) with direct access to the environment, whereas non-critical partitions are allowed a dynamic behaviour (e.g., allowing for dynamic memory management and automatic garbage collection) with access to the environment only through the hypervisor and/or hosting operating system.

Our approach, although sharing the same goal, is fundamentally different. We approach the problem from a language perspective, allowing the programmer to define applications with mixed criticality. The language design allows us to precisely define the set of allowed operations for each partition, and provides an outset for offline analysis. Given proper analysis and verified correctness of the tool-chain, separation can be achieved without the need of costly virtualisation and hypervisor techniques. Moreover, in our approach, interactions in and across regions of the system is controlled by the language semantics (amendable to analysis), free of references to any external mechanisms.

Ultimately, this will allow safe deployment of systems with mixed criticality even onto low-end microcontrollers such as the ARM Cortex M0/M3 family. In the following we will further discuss the outset for this endeavour.

### 3 Dynamic Semantics of RTFM-lang

In this section we informally discuss the dynamics semantics of RTFM-lang<sup>1</sup>. For the presentation we undertake the common notions of tasks(jobs) and resources used e.g., [5].

We associate each object (instance)  $o$  to a single unit resource  $r(o)$ , which must be claimed for the execution of a method  $o.m$ . Since the object state  $s(o)$  is completely encapsulated in the object  $o$  (we do not expose state variables directly in the interface, rather through set and get methods), the associated resource  $r(o)$  will protect the state  $s(o)$  from race conditions. Execution in the model is triggered by events (messages) on the form  $o.m(\dots)$ . Synchronous events are executed directly on behalf of the sender, while asynchronous events are dispatched by the run-time system. Asynchronous events either stem from the environment (e.g., typically as

<sup>1</sup>As a work in progress formal semantics will be defined in the  $\mathbb{K}$  framework.

a result of an interrupt), or from within the model (the `send` statement).

The static semantics ensures that the *critical* functions operate on the static object structure, with a static communication topology. This allows us to see execution in the system as a set of tasks  $T$ , where each task  $t \in T$  is a chain of synchronous events headed by a triggering asynchronous event. Looking at the grammar presented in Figure 1, we can see that the `send` statement takes an integer priority that is associated with the corresponding asynchronous event.

### 3.1 Interfacing the Environment

Interfacing with the environment is not explicitly defined in the presented grammar, but there are many options. One approach is that a system model requires a `Root` object implementing the set of interrupt handlers required by the underlying hardware (and RTFM-kernel). Notice that a priority  $p$  is associated with each interrupt handler.

```

1  class Root <> {
2      // LPC11U specific handlers
3      critical async 3 FLEX_INT0_IRQHandler() {
4          /* 0 - GPIO pin interrupt 0 */
5      }
6      critical async 2 FLEX_INT1_IRQHandler() {
7          /* 1 - GPIO pin interrupt 1 */
8      }
9      // ...
10     critical async 2 USBWakeup_IRQHandler() {
11         /* 30 - USB wake-up interrupt */
12     }
13     critical async 4 RTFM_Reset() {
14         /* Your program entry point */
15     }
16 }

```

Figure 5: Root class definition.

### 3.2 SRP Analysis and the RTFM-kernel

We now have sufficient information to perform SRP based analysis and scheduling of the static model. From the set of task  $T$  and set of objects  $O$ , a set of resources  $R$  with respective resource ceilings can be computed. This is sufficient information to perform SRP based scheduling. To this end, the RTFM-kernel has been developed to efficiently exploit the underlying interrupt hardware for making fixed priority scheduling decisions under SRP.

Moreover, the task model forms an outset for further analyses (e.g., response time, stack memory, and overall schedulability). To this end, additional requirements on inter-arrival times, WCETs, stack usage per task, etc. is of course required, and out of scope for this presentation.

## 4 Robust Scheduling of Mixed Critical Systems in RTFM-lang

Building from the discussion in SRP analysis and fixed priority scheduling by the RTFM-kernel, mixed critical models in the RTFM-lang can be robustly scheduled. Intuitively,

let critical functions (tasks) have priorities higher than non-critical functions. In this context, garbage collection (if so wished) can be added as a non-critical task. Under these conditions non-critical functions (and side effects thereof) will never preempt critical functions. This allows analysis (e.g., response time for critical functions) on the static model to hold in a mixed critical setting. W.r.t. blocking, a well know property of SRP is the bounded blocking to the single longest critical section a resource at same or higher priority is occupied. Remembering that resources are associated to an object  $o$ , and that a resource  $r(o)$  is claimed only for the period a  $o.m(\dots)$  is executing. Assuming that (subset) of methods exposed to non-critical functions is defined, the worst case blocking time (for each critical task) can be determined at compile time.

In order to ensure that the set of critical tasks can be determined at compile time, we can impose the restriction that non-critical tasks may never pend critical tasks. Instead, the pending is delegated to a critical method (called synchronously by the non-critical sender). This method takes the responsibility to determine whether to issue, queue or simply discard the job request. For the analysis to hold, the assumed minimum inter-arrival time must be obeyed. Other criteria, such as maximum queue length, message ageing, etc. can be implemented in such a high-level scheduler.

## 5 Conclusions and Future Work

In the is paper we have discussed the static and dynamic semantics of the experimental **RTFM**-language from the viewpoint of systems with mixed criticality. We have highlighted some outstanding features, such as the possibility to programmatically define and automatically verify critical functions to be free of dependencies to non-critical functions. This decoupling allows us, e.g., to freely deploy (potentially failing) dynamic memory allocations in non-critical functions, while still preserving safe operation for the critical functions. Moreover we have discussed the mapping from **RTFM**-language models to traditional notion of tasks and resources, and highlight the potential to SRP based techniques. In particular, fixed priority SRP scheduling under the **RTFM**-kernel would allow robust scheduling even under overload of non-critical functions.

## Acknowledgments

This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology), and the EU ARTEMIS JU funding, within project ARTEMIS/0001/2013, JU grant nr. 621429 (EMC2) and VINNOVA (Swedish Governmental Agency for Innovation Systems).

## 6 Disclaimers

The examples In Figures 1 to 3 given are only illustrative, in fact the mutual dependencies introduced would deadlock (and/or give rise to infinite execution). The Root example assumes (in order to release the lock on the Root object) that the handlers delegate work in terms asynchronous messages. Another abstraction allowing external bindings from within class definitions may prove a better approach, but out of scope for this presentation.

## References

- [1] ARTEMIS: Advanced Research & Technology for Embedded Intelligence and Systems, “Artemis strategic research agenda 2011.” <http://www.artemis-ia.eu/publication/download/publication/541>.
- [2] T. F. Serbanuta, A. Arusoai, D. Lazar, C. Ellison, D. Lucanu, and G. Rosu (2013), *The k primer (version 3.3)*, in Proceedings of International K Workshop (K’11), ENTCS, Elsevier. To appear.
- [3] G. Roşu (2014), *Matching logic: A logic for structural reasoning*, Tech. Rep. <http://hdl.handle.net/2142/47004>, University of Illinois.
- [4] Q. Xu, W. P. de Roever, and J. He (1997), *The rely-guarantee method for verifying shared variable concurrent programs*, Formal Asp. Comput., vol. 9, no. 2, pp. 149–174.
- [5] T. P. Baker (1990), *A stack-based resource allocation policy for realtime processes*, in proceedings of the Real-Time Systems Symposium, pp. 191–200, IEEE Computer Society.



# Handling Criticality Mode Change in Time-Triggered Systems through Linear Programming

**Mathieu Jan, Lilia Zaourar**

CEA, LIST, Embedded Real Time Systems Laboratory, F-91191 Gif-sur-Yvette, France; email: {Mathieu.Jan, Lilia.Zaourar}@cea.fr

**Vincent Legout**

Virginia Tech, Blacksburg, Virginia, USA; email: vlegout@vt.edu

**Laurent Pautet**

Institut Telecom, Telecom ParisTech, LTCI - UMR 5141 Paris, France; email: Laurent.Pautet@telecom-paristech.fr

## Abstract

*Mixed Criticality helps reducing the impact of pessimistic evaluation of Worst Case Execution Time for real-time systems. This is achieved by hosting low-criticality tasks on a same hardware architecture in addition to the classical high-critical tasks, when considering two-criticality levels. The Time-Triggered paradigm (TT) is a classical approach within industry to develop high-criticality tasks. Extending TT systems in order to integrate the support of MC scheduling therefore requires the generation of two schedule tables, one for each criticality level. However, a switch between the schedule tables must not lead to an unschedulable situation for the high-criticality tasks. In this work, we show how a linear programming approach can be used to generate these schedule tables in a consistent way for dual-critical problems on multiprocessor architectures.*

**Keywords:** Real-time scheduling, Time-Triggered, mixed-criticality.

## 1 Introduction

Industrial fields, such as automotive [1] or control automation [2], consider the Time-Triggered [3] (TT) paradigm as a solution to build hard real-time systems. In the TT paradigm, the tasks are triggered by the advancement of time. The scheduling decisions are usually computed off-line and made available to the Real-Time Operating System (RTOS) through a schedule table. While the TT paradigm provides a predictable execution, the static scheduling approach is considered to lead to a poor resource utilization in the average case. The design of TT systems and the associated schedulability demonstrations must indeed be performed in the worst-case situation. These unused processing capabilities motivate the adding of Mixed-Criticality (MC) scheduling techniques within TT systems [4].

The goal of MC scheduling is to increase the schedulability of the low-criticality tasks, while still guaranteeing in the worst-case scenario the schedulability of the high-criticality tasks. In a previous work, we focused on the use of the elastic task model to include MC scheduling within TT systems [5]. In this work, we rely on the task model mainly used within the MC scheduling community [6], called the Vestal task model. This task model extends the classical periodic task model with: 1) two Worst-Case Execution Time (WCET) values, named  $C_i(LO)$  and  $C_i(HI)$ , and 2) a criticality level  $\chi_i$ , which can be either  $LO$  or  $HI$ . Then, two execution modes are assumed, namely  $HI$  and  $LO$ , and the system starts in the  $LO$  mode.  $C_i(LO)$  is the maximum allowed execution time for the task in the  $LO$  mode, while  $C_i(HI)$  is the maximum allowed execution time for the task in the  $HI$  mode. For the  $HI$ -criticality tasks we have  $C_i(LO) < C_i(HI)$  and for the  $LO$ -criticality tasks  $C_i(LO) = C_i(HI)$ . The rationale is that the higher the criticality level is, the more conservative the verification process is and hence the greater the WCET value is. Whenever a  $HI$ -criticality task exceeds its assigned  $C_i(LO)$  value, the system switches to the  $HI$  mode. In this mode, only the schedulability of the  $HI$ -criticality tasks is ensured, assuming  $C_i(HI)$  for the WCET values.

Extending TT systems to cope with MC scheduling requires the definition of two schedules tables, named  $S_{LO}$  and  $S_{HI}$ .  $S_{LO}$  (resp.  $S_{HI}$ ) is used while the system is in the  $LO$  (resp.  $HI$ ) mode. [4] has stated the TT schedulability conditions that apply on these schedules for dual-criticality MC instances of task sets. The main issue is to guarantee that a mode change from  $S_{LO}$  to  $S_{HI}$  cannot lead to an unfeasible schedule for the  $HI$ -criticality tasks, i.e. the remaining time is not sufficient to completely schedule all the  $HI$ -criticality tasks.  $S_{HI}$  is indeed concerned by the schedulability of the  $HI$ -criticality tasks but should be built so that the schedulability of the  $LO$ -criticality tasks is maximized in  $S_{LO}$ . Building  $S_{LO}$  and  $S_{HI}$  cannot therefore be made independently in order to improve the resource utilization in the average case.

We propose two approaches to build  $S_{LO}$  and  $S_{HI}$  for (dual-criticality) instances of MC task sets. Both are based on the use of a linear programming approach. The remainder of this paper is as follows. Section 2 describes the related work. Section 3 formulates our linear programs to handle the criticality mode change in TT scheduling. Section 6 provides a first analysis of our solutions and section 7 concludes.

## 2 Related work

Note that existing work focuses on finite set of jobs whose exact arrival times are known a priori, as the results can be easily extended in order to address TT systems. The proposed algorithm must indeed only be applied over the hyper-period of the periodic task set being considered.

The first work on using MC scheduling within TT systems [4] studied how to generate  $S_{LO}$  and  $S_{HI}$  that can correctly schedule a MC job set modeled using the Vestal task model. It was inspired by the mode-change approach used to increase the flexibility of the TT scheduling in [7]. As the TT schedulability of MC tasks is NP-hard in the strong sense, they propose a polynomial-time algorithm for building  $S_{LO}$  and  $S_{HI}$  that is sufficient but not necessary. That is, the algorithm can fail to generate such tables for schedulable MC job sets, but if it succeeds then tables can correctly schedule them. The algorithm first computes a total priority ordering of the jobs using the Own-Criticality Based Priority (OBCP) algorithm [8]. Based on this priority ordering,  $S_{LO}$  is first built assuming  $C_i(LO)$  for all the jobs. Then,  $S_{HI}$  is generated assuming this time  $C_i(HI)$  for all the jobs (we remind that when  $\chi = LO$ ,  $C_i(LO) = C_i(HI)$ ).

[9] introduces a much more elaborated algorithm to build at the same time  $S_{LO}$  and  $S_{HI}$  assuming a slot scheduling approach.  $HI$ -criticality jobs are first splitted into two jobs noted  $J_i^{LO}$  and  $J_i^{\Delta}$ .  $J_i^{LO}$  represents the  $C_i(LO)$  of that job in the  $LO$  mode, while  $J_i^{\Delta}$  represents the additional WCET assumed when in the  $HI$  mode (i.e.  $\Delta_i = C_i(HI) - C_i(LO)$ ). Release time and deadline of these sub-jobs are computed so that each job has a maximum interval for its execution. A precedence constraint between sub-jobs is added in order to ensure a correct execution. Then, the proposed algorithm uses an heuristic to explore the set of possible scheduling decisions represented as a tree search. Based on the demand of  $HI$ -criticality jobs, a backtracking heuristic is used to cut from the tree search paths leading to unfeasible schedules.

[10] focuses on adding MC scheduling support within TT legacy systems, where the existing schedule table is considered to be  $S_{HI}$ . The proposed algorithm extends the slot-shifting based scheduling [11] in order to keep track of the spare capacities in each interval for both the  $HI$  and the  $LO$ -criticality jobs (named  $sc^{HI}$  and  $sc^{LO}$  respectively). A negative spare capacity means that some execution time must be borrowed from the other slots. If  $sc^{LO} < 0$  then a  $HI$ -criticality job has exceeded its  $C_i(LO)$  value and therefore only  $HI$ -criticality jobs must be executed. Finally, the legacy TT schedule is used only when  $sc^{HI} = 0$ . Note that this legacy TT schedule can prevent  $S_{LO}$  to be build, while [9] or [4] could produce a correct  $S_{LO}$ . It is the price to pay

to avoid additional certification costs by keeping unchanged  $S_{HI}$ .

When considering TT systems, the previously introduced algorithms are called Single Time Table per Mode (STTM). [12] proves that the STTM approach dominates MC scheduling algorithms that define the priorities of jobs depending on the criticality mode (called FPM for Fixed Priority per Mode). They propose an algorithm in order to transform a FPM priority assignment into a set of STTM tables.

In this paper, the objectives of our contributions aim at revisiting these approaches for TT MC systems using Linear Programming (LP) techniques.

## 3 Problem Description using LP Approach

We first introduce the task model and notations we use in the remainder of this paper, before presenting our two linear programming approaches for building  $S_{LO}$  and  $S_{HI}$ .

As stated in the introduction, we rely on the Vestal task model and consider only two criticality levels, a restriction often assumed in MC scheduling [13]. We only state additional notations not introduced in the previous sections. Let  $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$  be a set of  $n$  independent, synchronous, preemptible and implicit deadline tasks. Tasks can migrate from one processor to another. We let  $M$  denote the number of processors. Each task  $\tau_i \in \Gamma$  has the following temporal parameters  $\tau_i = (\chi_i, P_i, C_i(LO), C_i(HI))$  with  $P_i$  the period of the task. Let  $H$  be the hyper-period of the task set. It is equal to the least common multiple of all periods of tasks in  $\Gamma$ . As in [14], the hyper-period  $H$  is divided in intervals, an interval being delimited by two task releases. We let  $n_{HI}$  denote the number of  $HI$ -criticality tasks and  $n_{LO}$  the number of  $LO$ -criticality tasks (thus  $n_{HI} + n_{LO} = n$ ).

A job  $j$  can be present on several intervals and  $E_i$  is the set gathering these intervals. We let  $w_{j,k}$  denote the weight of job  $j$  on interval  $k$ . We denote by  $I$  the set of intervals and  $|I_k|$  the duration of the  $k^{th}$  interval.  $J_k$  is the set of jobs within interval  $k$ . The weight of each job is the amount of processor necessary to execute job  $j$  on interval  $|I_k|$  only (it is not an execution time but a fraction of it).  $J_{\Gamma}$  is the job set of all jobs of  $\Gamma$  scheduled during the hyper-period  $H$ .

A job  $j_{LO}$  represents an instance of a  $LO$ -criticality task, while  $j_{HI}$  is an instance of a  $HI$ -criticality task.  $J_{LO}$  and  $J_{HI}$  are the job sets of respectively all the  $LO$  and the  $HI$ -criticality jobs from  $\Gamma$ . We let  $w_{j,k}^{LO}$  denote the weight of job  $j$  in interval  $k$  in  $S_{LO}$ , while  $w_{j,k}^{HI}$  is the weight of job  $j$  in interval  $k$  in  $S_{HI}$ . A RTOS is used to detect when a  $j_{HI}$ , i.e.  $HI$ -criticality job, exceeds its  $C_i(LO)$  value.

Finally, note that our approach to build  $S_{HI}$  and  $S_{LO}$  is based on the use of LP to compute off-line the weights of each job on all intervals. Then, within an interval either a dynamic or static approach can be used to schedule jobs. As we are considering the TT approach, in this work we assume that the triggering of jobs is also computed off-line, for instance by using McNaughton's algorithm [15]. These scheduling decisions are also stored in  $S_{HI}$  and  $S_{LO}$ . In this paper, we do not focus on this part of our scheduling approach.

## 4 LP scheduling for HI-mode first: LPMC-HI

In our first proposal,  $S_{HI}$  and  $S_{LO}$  are built in two separate steps, although the objective used when building  $S_{HI}$  prepares the computation of  $S_{LO}$ . We express this solution as two linear programs, one for each table to build, and we name it LPMC-HI for *Linear Programming for Mixed-Criticality HI-mode first*. However, the constraints of the first linear program are the schedulability of  $J_{HI}$  tasks, while its objective is to optimize the schedulability of  $J_{LO}$  tasks in HI-mode. This prepares the input, i.e. the  $w_{j,k}^{LO}$  for the jobs from  $J_{HI}$ , for the second linear programs dealing with the LO-mode.

First, we focus on building  $S_{HI}$ . The classical temporal schedulability constraints [14] are expressed to compute the optimal job weights on each interval for all the jobs from  $J_{\Gamma}$ . First, the sum of all job weights on an interval must not exceed the processor maximum capacity:

$$\sum_{j \in J_k} w_{j,k}^{HI} \leq M, \forall k \in I \quad (1)$$

Then, the weight of each job must not exceed each processor maximum capacity (no parallel jobs):

$$0 \leq w_{j,k}^{HI} \leq 1, \forall k \in I, \forall j \in J_{\Gamma}. \quad (2)$$

Finally, we express two different constraints for the completion of jobs from  $J_{LO}$  and  $J_{HI}$ . First, only the schedulability of the jobs from  $J_{HI}$  has to be ensured and therefore must be completely executed:

$$\sum_{k \in E_j} w_{j,k}^{HI} \times |I_k| = C_i(HI), \forall j \in J_{HI}. \quad (3)$$

Second, the schedulability of the jobs from  $J_{LO}$  may not be ensured while in  $S_{HI}$ . This means that some jobs from  $J_{LO}$  may not be completely executed:

$$\sum_{k \in E_j} w_{j,k}^{HI} \times |I_k| \leq C_i(LO), \forall j \in J_{LO}. \quad (4)$$

As far as  $S_{HI}$  is concerned, our objective is to prepare the building of  $S_{LO}$  in order to maximize the schedulability of  $J_{LO}$ , while still guaranteeing in the worst-case scenario the schedulability of  $J_{HI}$ . To this end, we introduce a decision variable to account when a job from  $J_{LO}$  has been completely executed, i.e.  $\sum_{k \in E_j} w_{j,k}^{HI} \times |I_k| = C_i(LO)$ . Let  $F_j$  be this decision variable that is equal to 1 if the job  $j$  from  $J_{LO}$  has been completely executed, and 0 otherwise. Then, our objective function can be defined as follows:

$$\text{Maximize } \sum_{j \in J_{LO}} F_j \quad (5)$$

This objective function computes the weights of a schedule in which a maximum number of jobs from  $J_{LO}$  are completely executed, while ensuring the schedulability for jobs from  $J_{HI}$ .

We now focus on the LO-mode. For building  $S_{LO}$  we have to compute  $w_{j,k}^{LO}$  for each job from  $J_{\Gamma}$  assuming that its WCET

is equal to  $C_i(LO)$ . That is the execution time of each job  $j$  from  $J_{HI}$  is reduced by  $\Delta_i$  (see section 2). For each job  $j$  in  $J_{HI}$ ,  $w_{j,k}^{LO}$  is equal to  $w_{j,k}^{HI}$  till the  $C_i(LO)$  is not exceeded. This differs from [4], where at a mode change, no scheduler could have given more time to the HI-criticality jobs than the proposed algorithm. In LPMC-HI, these jobs can be delayed in order to completely execute, over a set of intervals, some LO-criticality jobs.

Next, we have to compute  $w_{j,k}^{LO}$  for all the jobs from  $J_{LO}$ . While  $S_{HI}$  was generated with a maximum number of jobs from  $J_{LO}$  completely executed, our second linear program could only compute the weights of the jobs from  $J_{LO}$  that have not yet been scheduled. However, we believe this reduces the search space when building  $S_{LO}$ , as we remind that only jobs from  $J_{HI}$  must be scheduled in  $S_{HI}$ . While a reduced search space seems an interesting property, as it decreases the execution time required for solving the linear program, we believe it also reduces the schedulability bound that can be achieved. To compute  $w_{j,k}^{LO}$  for all the jobs from  $J_{LO}$ , the classical temporal constraints only have to be modified in order to use  $w_{j,k}^{LO}$  for all the jobs from  $J_{HI}$  as fixed values and not as variables. In the next equation, the value of a variable  $w$  is noted  $w'$  to depict this point :

$$\sum_{j_{LO} \in J_k} w_{j,k}^{LO} + \sum_{j_{HI} \in J_k} w_{j,k}^{LO'} \leq M, \forall k \in I \quad (6)$$

$$0 \leq w_{j,k}^{LO} \leq 1, \forall k \in I, \forall j \in J_{LO}. \quad (7)$$

$$\sum_{k \in E_j} w_{j,k}^{LO} \times |I_k| = C_i(LO), \forall j \in J_{LO}. \quad (8)$$

Note that this second LP has no objective function as any feasible solution given by the solver generates a valid scheduling.

LPMC-HI can lead to situations where  $S_{LO}$  cannot be computed. The jobs from  $J_{HI}$  are indeed concentrated in some particular intervals in  $S_{HI}$  and then their total weights are simply reduced over these intervals to match their lower  $C_i(LO)$ . However, redistributing the weights of jobs from  $J_{HI}$  while computing  $S_{LO}$  would increase the schedulability bound that can be achieved for the jobs from  $J_{LO}$ . Section 6 illustrates this point using an example.

## 5 LP scheduling for both LO- and HI-modes: LPMC-Both

In our second approach, we explore such an alternative strategy for computing the weights in order to improve the success ratio of the scheduling. We thus consider the generation of  $S_{LO}$  and  $S_{HI}$  at the same time, i.e. within the same linear program, and therefore name this approach *LPMC-Both*. We split each HI-criticality job into two sub-jobs:  $j_{LO}$  and  $j_{\Delta}$  and consider  $j_{LO}$  as a LO-criticality job that we added in  $J_{LO}$ . A precedence constraint will be expressed later to ensure building correct schedules. LPMC-Both is similar to [9] and we therefore use the same notations as in this work (see sect. 2). Additionally, we let  $w_{j,k}^{\Delta}$  denote the weight of a job  $j_{\Delta}$ .

A first set of constraints must be expressed for  $S_{HI}$  in order to ensure the schedulability of all the jobs from  $J_{HI}$ . This is similar to the equations (1), (2) and (3). The only difference is that now the weight of each job in  $S_{HI}$  is defined as follows:

$$w_{j,k}^{HI} = w_{j,k}^{LO} + w_{j,k}^{\Delta}, \forall k \in I, \forall j \in J_{HI} \quad (9)$$

Precedence constraints are then required to ensure a correct schedule of each job from  $J_{HI}$ , that is the  $w_{j,k}^{\Delta}$  must be null till  $\sum_{m=1}^k w_{j,m}^{LO} \times |I_m| \leq C_i(LO)$ . This prevents sub-jobs  $j_{LO}$  and  $j_{\Delta}$  to be present in the same interval in  $S_{LO}$ . Avoiding this situation ensures that a criticality mode change from  $S_{LO}$  to  $S_{HI}$  is possible, i.e. that it does not lead to an unfeasible schedule, at every point where all the jobs from  $J_{HI}$  can first exceed their  $C_i(LO)$  values. This corresponds to the *switch through property* described in [7] for these points. Note that this property is ensured in our first scheduling approach by how we compute  $w_{j,k}^{LO}$  for each job  $j$  from  $J_{HI}$ . In the first interval  $k$  in which a job  $j_{HI}$  exceeds its  $C_i(LO)$  value, note that the two sub-jobs  $j_{LO}$  and  $j_{\Delta}$  can be present. However, as  $w_{j,k}^{HI}$  cannot be higher than 1 (eq. 2), the weight left to  $j_{\Delta}$  in interval  $k$  is constrained so that a schedule where  $j_{LO}$  and  $j_{\Delta}$  cannot be executed in parallel can be found (i.e.  $w_{j,k}^{\Delta} + w_{j,k}^{LO} \leq |I_k|$ ). Finally, in the other intervals the solver has no constraint for computing  $w_{j,k}^{\Delta}$ .

Then, a second set of constraints must be expressed for  $S_{LO}$  in order to ensure the schedulability of all the jobs from  $J_{LO}$ . These constraints are identical to the equations (7) and (8), in addition to the following constraint:

$$\sum_{j \in J_k} w_{j,k}^{LO} \leq M, \forall k \in I \quad (10)$$

Finally, we use the same objective function as (5), that is maximize the number of schedulable jobs from  $J_{LO}$ . It therefore requires the same decision variable to account when a job from  $J_{LO}$  has been completely executed. In the end, if a solution can be found, then the output of LPMC-Both is the weights of each job to be used to build both  $S_{LO}$  and  $S_{HI}$ .

## 6 First analysis of LPMC-HI and LPMC-Both

We first compare LPMC-HI and LPMC-Both in terms of complexity. We first focus on LPMC-HI. The total number of decision variables in the first LP of LPMC-HI is equal to  $|I| \times n$  for the weights of all jobs, plus  $|J_{LO}|$  for the  $F_j$ . In the second LP of LPMC-HI, this number is reduced to  $|I| \times n_{LO}$  as only the weights of jobs from  $J_{LO}$  are computed when building  $S_{LO}$ . In the first (resp. second) LP of LPMC-HI, the number of constraints is equal to its number of variables plus  $|I| + |J_{\Gamma}|$  (resp.  $|I| + |J_{LO}|$ ) due to the equations (1), (3) and (4) (resp. (6) and (8)). We now focus on LPMC-Both. Compared to LPMC-HI, the total number of decision variables in LPMC-Both is increased by  $2 \times |I| \times n_{HI}$ . This comes from additional weights introduced by the job splitting and for implementing the precedence constraints. The number of constraints of LPMC-Both is equal to the sum of:  $|I| \times (n+1) + |J_{\Gamma}|$  for computing  $S_{LO}$ ,  $|I| \times (n_{HI}+1) + |J_{HI}|$

	$\chi_i$	$P_i$	$C_i(LO)$	$C_i(HI)$
$\tau_1$	LO	2	1.5	1.5
$\tau_2$	HI	4	2	3
$\tau_3$	HI	3	1	2

Table 1: Task set with  $\tau_1$  a LO-criticality task.

for computing  $S_{HI}$  and  $2 \times |I| \times n_{HI}$  for dealing with the precedence constraints. The complexity of LPMC-Both is therefore higher than the complexity of LPMC-HI. However, the computational complexity of LPMC-HI and LPMC-Both depends on the number of intervals, which is limited in industrial configurations usually showing harmonic periods ( $[1,2]$ ).

We now compare both approaches in terms of efficiency. Table 1 depicts a task set running on a dual-core ( $M = 2$ ) and made of three tasks where  $\tau_1$  is a LO-criticality task. Figure 1 shows  $S_{HI}$  computed by LPMC-HI. The third and sixth instances of  $\tau_1$  cannot be completely executed in the intervals  $I_4$  and  $I_8$ , leading to  $F_1 = 4$  (out of 6). Note that the second and fifth instances of  $\tau_1$  span over 2 intervals, i.e. respectively  $I_2, I_3$  and  $I_6, I_7$ . The other instances require only 1 interval. When trying to compute the corresponding  $S_{LO}$ ,  $w_{3,4}^{LO}$  is equal to 0.5, as the  $C_i$  of the second instance of  $\tau_3$  is reduced by 1 unit of time in interval  $I_4$ . However, the third instance of  $\tau_1$  cannot be scheduled as  $w_{1,4}^{LO}$  should be equal to 0.75 in order to satisfy the equation (8). But then, the equation (6) would not be satisfied as the utilization would be equal to 2.5 and hence higher than  $M$ . A valid  $S_{LO}$  cannot therefore be computed. As shown by Figure 2, both  $S_{LO}$  and  $S_{HI}$  can be computed using LPMC-Both thanks to its ability to distribute the weights over all the intervals.

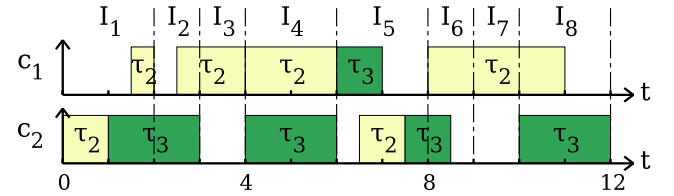


Figure 1: Possible  $S_{HI}$  for the task set of table 1 computed by the LPMC-HI leading to an unfeasible  $S_{LO}$ .

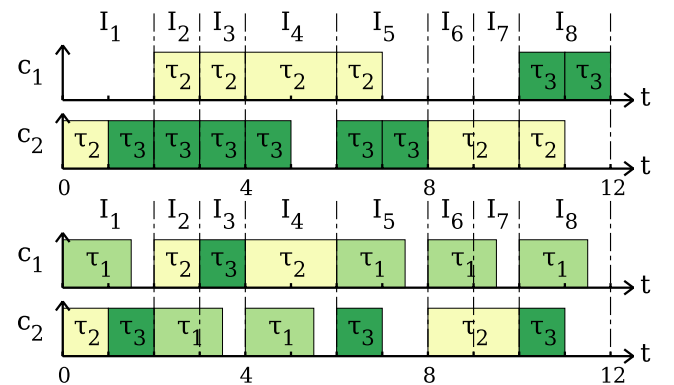


Figure 2: Possible  $S_{HI}$  (top) and  $S_{LO}$  (bottom) for the task set of table 1 computed by LPMC-Both.

## 7 Conclusion

The Time-Triggered (TT) paradigm is one solution used within industrial fields to design hard real-time systems subject to certification constraints. While the TT paradigm provides interesting properties, such as determinism, this comes at the price of low resource utilization in the average case. Mixed-Criticality (MC) scheduling aims at providing an efficient use of the processing capabilities available in the average case through the execution of low-criticality tasks, while ensuring schedulability for the high-criticality in the worst-case.

TT relies on a off-line computation of scheduling decisions made available at run-time through a schedule table. In this work, we consider dual-critical problems requiring the construction of two schedule tables. The main difficulty when building them is to ensure that switching from the *LO* table to the *HI* table is possible, i.e. does not lead to unfeasible schedules when a *HI* criticality task exceeds its *LO* behaviour. We propose two approaches, named LPMC-HI and LPMC-Both, based on the use of linear programs to build these tables. We are currently implementing them in order to evaluate their success ratio in scheduling of job sets whose utilizations are uniformly distributed, as in [9]. In future work, we plan to integrate additional constraints in the generation of TT schedule, such as energy consumption as presented in [16].

## References

- [1] D. Chabrol, D. Roux, V. David, M. Jan, M. A. Hmid, P. Oudin, and G. Zeppa (2013), *Time- and angle-triggered real-time kernel for powertrain applications*, in Proc. of the Design, Automation & Test in Europe Conf. (DATE), (Grenoble, France), pp. 1060–1063.
- [2] M. Jan, V. David, J. Lalande, and M. Pitel (2010), *Usage of the safety-oriented real-time OASIS approach to build deterministic protection relays*, in Proc. of the 5<sup>th</sup> Intl. Symp. on Industrial Embedded Systems (SIES 2010), (Trento, Italy), pp. 128–135.
- [3] H. Kopetz 1998, *The time-triggered model of computation*, in Proc. of the Real-Time Systems Symp. (RTSS), (Madrid, Spain), pp. 168–177.
- [4] S. Baruah and G. Fohler (2011), *Certification-cognizant time-triggered scheduling of mixed-criticality systems*, in Proc. of the 32nd Real-Time Systems Symp. (RTSS), (Vienna, Austria), pp. 3–12.
- [5] M. Jan, L. Zaourar, and M. Pitel (2013), *Maximizing the execution rate of low-criticality tasks in mixed criticality systems*, in Proc. of the 1st Intl. Workshop on Mixed Criticality Systems (WMC), (Vancouver, Canada), pp. 43–48.
- [6] S. Vestal (2007), *Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance*, in Proc. of the 28th Real-Time Systems Symp. (RTSS), (Tucson, USA), pp. 239–243.
- [7] G. Fohler (1993), *Changing operational modes in the context of pre run-time scheduling*, vol. E76-D, no. 11, pp. 1333–1340.
- [8] S. Baruah, H. Li, and L. Stougie (2010), *Towards the design of certifiable mixed-criticality systems*, in Proc. of the 16th Intl. Conf. on Real-Time and Embedded Technology and Applications Symp. (RTAS), (Stockholm, Sweden), pp. 13–22.
- [9] J. Theis, G. Fohler, and S. Baruah (2013), *Schedule table generation from time-triggered mixed criticality systems*, in Proc. of the 1st Intl. Workshop on Mixed Criticality Systems (WMC), (Vancouver, Canada), pp. 79–84.
- [10] J. Theis and G. Fohler (2013), *Mixed criticality scheduling in time-triggered legacy systems*, in Proc. of the 1st Intl. Workshop on Mixed Criticality Systems (WMC), (Vancouver, Canada), pp. 73–78.
- [11] G. Fohler (1995), *Joint scheduling of distributed complex periodic and hard aperiodic tasks in statically scheduled systems*, in Proc. of the 16th Real-Time Systems Symp. (RTSS), (Pisa, Italy), pp. 152–161.
- [12] D. Succi, P. Poplavko, S. Bensalem, and M. Bozga (2013), *Time-triggered mixed-critical scheduler*, in Proc. of the 1st Intl. Workshop on Mixed Criticality Systems, (Vancouver, Canada), pp. 67–72.
- [13] S. K. Baruah, V. Bonifaci, G. D’Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie (2012), *Scheduling real-time mixed-criticality jobs* IEEE Trans. Computers, vol. 61, no. 8, pp. 1140–1152.
- [14] M. Lemerre, V. David, C. Aussaguès, and G. Vidal-Naquet (2008), *Equivalence between schedule representations: Theory and applications* in Proc. of the Real-Time and Embedded Technology and Applications Symp., (St. Louis, USA), pp. 237–247.
- [15] R. McNaughton (1959), *Scheduling with deadlines and loss functions*, Management Science, vol. 6, pp. 1–12.
- [16] V. Legout, M. Jan, and L. Pautet (2013), *Mixed-criticality multiprocessor real-time systems: Energy consumption vs deadline misses*, in Proc. 1st workshop on Real-Time Mixed Criticality Systems, (Taiwan).

# Mixed Criticality over Switched Ethernet Networks

*Olivier Cros, Frédéric Fauberteau, Xiaoting Li*

*ECE Paris, 37, quai de Grenelle, 75015 Paris, France; email: {cros, fauberte, xiali}@ece.fr*

*Laurent George*

*University of Paris-Est, LIGM, Cité Descartes, Bâtiment Copernic - 5, bd Descartes, 77454 Champs sur Marne, France; email: lgeorge@ieee.org*

## Abstract

*In this paper, we focus on real-time switched Ethernet networks with mixed-criticality constraints. We are interested in (i) exploiting IEEE 1588 Precision Time Protocol (PTP) to implement criticality propagation techniques in such networks and (ii) analyzing delay of criticality switching. This work presents how to integrate criticality concepts for messages sent on Ethernet networks using PTP protocol. Concerning the delay of criticality switching, we consider FIFO and Fixed-Priority scheduling policies.*

*Keywords: Real-Time, Ethernet, IEEE 1588, scheduling, mixed criticality.*

## 1 Introduction

In recent years, a new development of real-time scheduling theory has been proposed with the introduction of Mixed-Criticality (MC) systems, in which a real-time system can be certified with various levels of assurance function of the criticality levels it may run. In this paper, we show that MC can also be a useful paradigm in real-time systems for the dimensioning of switched Ethernet networks where message payload or message inter-arrival time can change according to the criticality level of the system. This is of special interest in intelligent transportation systems where different situations can happen having different criticality levels. For examples, a video surveillance system, installed on board a bus, will result in longer frames been sent if a problem during transport is detected, to send better image resolution. Another situation could also happen where shorter message inter-arrival times would be necessary to send more often a critical information (for e.g. speeding, opened door, ...). The problem of MC management in network context lies on how to propagate the information of criticality through the nodes. To solve this problem, we propose to integrate criticality management information into IEEE 1588 Precision Time Protocol (PTP) clock synchronization protocol frames. We study the impact of this integration in terms of delays and performance.

## 1.1 Related work

The concept of MC has been introduced by Vestal [1] in 2007 to tackle this problem. A system is said to be mixed-critical if it has a number of functionalities with different criticalities. In Vestal's interpretation of MC, the correctness and timeliness of tasks that are more critical must be guaranteed with a higher degree of confidence than that of tasks that are less critical. To achieve this varying level of confidence, the method used to determine the worst case execution time (WCET) of a task can vary according to the criticality level they are allowed to run (we consider two criticality levels, LO and HI in this paper). If the criticality level is HI, only highly critical tasks are allowed to run. For tasks in HI mode, a very conservative tool should be used to compute their pessimistic WCETs, to be sure that no WCET overrun happens at run time. The reason why higher WCETs are synonymous with higher confidence is because the probability that during operation a program will exceed its WCET is lower if the WCET is higher. This important over-dimensioning is required for the certification issues of Certification Authorities (CA), and is acceptable if only limited to high critical tasks. When the system runs in LO criticality mode, LO and HI tasks are allowed to run as long as they do not execute for a duration higher than or equal to their optimistic WCET defined in LO mode. For instance a measurement based approach can be used to determine the WCET of tasks in LO mode. This can lead at run time to WCET overruns that must be handled by the system. A HI task can run in both modes LO and HI, a LO task can only execute in LO mode.

As shown in [2], the problem of MC scheduling is highly intractable, even for level-two MC. Nevertheless, two techniques for scheduling MC systems are proposed in [3]. Then, the algorithm EDF-VD is introduced in [4] to address the problem of MC with EDF scheduling for implicit deadlines. This work has been extended in [5]. A response-time analysis for MC systems has also been presented in [6]. EDF based on virtual deadline has then been generalized by the Ekberg and Yi's method [7] for arbitrary deadlines.

Few work has been done in the context of networked systems. In the context of CAN networks, the authors of [8] present a MC protocol for CAN networks and provide a sufficient response-time analysis and an optimal priority assignment. In



the context of FlexRay networks, a framework for automatic schedule synthesis has been proposed [9]. The respect of the timing requirements has been formulated as an Integer Linear Programming. In the context of public transport, an architecture to provide interoperability in the communication systems of a bus has been proposed [10]. In order to adapt MC systems in such an architecture, an adapted model has been presented to take into account the non-preemptive network flows. In the context of the Wireless Multimedia Sensor Networks, a MC scheduling scheme has been proposed to improve the transmission of image data stream over a WiFi network [11].

The manufacturers are more and more interested in the MC systems. For example, several companies as STMicroelectronics, Thales or TTTech has joined the DREAMS European project [12]. The objective of this project is to develop a cross-domain architecture and design tools for networked complex systems where application subsystems of different criticality, executing on networked multi-core chips, are supported.

We use the trajectory approach to compute end-to-end delays in a switched Ethernet network [13]. Although this approach has been shown optimistic in some corner cases [14], a recent work [15] provide a review of the identified problems and propose corrections.

## 1.2 Addressed points

We organize the paper as follows: we introduce MC in switched Ethernet networks by considering two main points. The first one considers the concrete integration of MC on a switched Ethernet network. We revisit PTP time-synchronization protocol to answer our first question: **How to manage MC over a switched-Ethernet network?**

The second point studies the time needed to switch from one criticality level to another in a switched Ethernet network. We consider the delay taken by a network to transmit the criticality-level information and the criticality switching request to all of its nodes (starting from a given master node). This allows us to answer to the second question: **What is the maximum time needed to change the criticality in switched-Ethernet network?**

## 2 Mixed criticality in a switched Ethernet network

The point that is important to focus on in a network context is the transmission delay of the information. Indeed, in classical mono or multiprocessor, we can easily make the hypothesis that transmitting a criticality information from one processor to another is null in terms of delay. In a network context, transmitting an information implies a latency, due to physical link transmission time. In this paper, we consider two criticality modes denoted LO and HI. The principles presented in this paper can be extended to more than two criticality levels. For the sake of simplicity, we only consider two criticality levels.

### 2.1 Network model

We consider a network denoted  $\mathcal{N}$  as a set of nodes consisting of  $e$  End Systems (ES) and  $s$  Switches (S). The whole network can be denoted  $\mathcal{N} = (\{ES_1, \dots, ES_e\}, \{S_1, \dots, S_s\})$ . We represent in Figure 1 a example consisting of 6 ES connected to 3 switches.

We consider the case of a statically defined Ethernet network with a set of flows following a dedicated path of switches. We consider a configuration similar to an AFDX network. We will consider a more general network as a further work.

The inputs and outputs of the network are source nodes (*i.e.* the ES). These ES are interconnected by a full duplex switched Ethernet. Links between switches are full-duplex, which guarantees no collisions on links. We consider a homogeneous single network.

Each source node sends a set of flows through an output port with a buffer supporting First In First Out (FIFO) or Fixed-Priority (FP) scheduling. A node can be connected to only one port of a switch and each port of a switch can be connected to at most one node. Each switch uses a store and forward policy. In each Virtual-LAN (VLAN), there is one buffer per output port which supports FIFO or FP scheduling and receives frames from input ports and forwards them to the corresponding output ports based on a static routing table.

There is a switching latency (technological latency) to deal with the frame forwarding between an input port and an output port of a given switch and it is upper bounded by a known value  $sl$ .

An example of architecture that we consider is depicted in Figure 1. It includes six End Systems  $ES_1, ES_2, ES_3, ES_4, ES_5$  and  $ES_6$  interconnected by three switches  $S_1, S_2$  and  $S_3$  via full-duplex links and  $ES_6$  is a sink node.

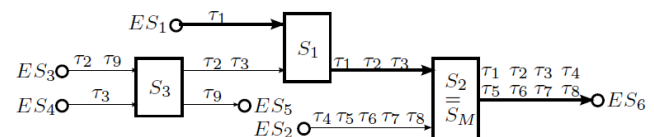


Figure 1: Network architecture

A switch is supposed to be able to take into account HI flow characteristics (Worst Case Transmission Time and inter-arrival time). A static table associated to the HI flows sent by a switch is stored in each switch. The criticality of a switch can evolve (i) as a function of flow characteristics it receives or (ii) when receiving specific PTP messages, modified to support criticality propagation. The characterization of a PTP frame is given in more details in Section 3.

In our work, we consider two kinds of switches following the IEEE 1588 PTP protocol: (i) one master criticality-management switch denoted  $S_M$  and (ii) slave switches. We denote  $S_k$  a switch of index  $k$ . The master criticality-management switch  $S_M$  is the last visited switch (switch  $S_2$  in figure 1). It is in charge of propagating criticality changes to all ES and to all slave switches. This is done:



- from *LO* mode to *HI*: if at least on  $S_M$ , one HI flow has characteristics changing from LO to HI mode (WCTT and inter-arrival time),
- from *HI* mode to *LO*: if all HI flows on  $S_M$  have characteristics changing from HI to LO mode.

## 2.2 Flow model

In order to understand mixed-criticality management in switched Ethernet networks, we need to define a set of notations that will be used in the paper to describe a flow.

We consider a set of network flows that we denote  $\tau = \{\tau_1, \dots, \tau_n\}$  a set of  $n$  network flows. The set  $\tau^{LO}$  (respectively  $\tau^{HI}$ ) denotes LO (respectively HI) flows in  $\tau$ .

A flow  $\tau_i \in \tau^{HI}$  is characterized by: (i) a set of Worst-Case Transmission Times (WCTT) denoted  $\{C_i^{LO}, C_i^{HI}\}$  and (ii) a set of associated minimum inter-arrival times (or period) denoted  $\{T_i^{LO}, T_i^{HI}\}$  respectively in LO and HI criticality.

From this definition, no constraint is set on the values of WCTT and inter-arrival times of HI flows in LO and HI modes except that a switch should be able to detect HI flow evolutions, i.e. we suppose that either  $C_i^{LO} \neq C_i^{HI}$  OR  $T_i^{LO} \neq T_i^{HI}$ .

A flow  $\tau_i \in \tau^{LO}$  is only defined by  $(C_i^{LO}, T_i^{LO})$ .

The path of a frame of a flow  $\tau_i$  from a switch  $S_x$  to a switch  $S_y$  is denoted  $\mathcal{P}_{xy}^i$ . The path between a switch  $S_k$  and the master switch  $S_M$  is denoted  $\mathcal{P}_{kM}^i$ .

Changing the criticality level in the network is therefore based on WCTT and/or worst case inter-arrival time flow observations.

If any switch in the network detects that a HI frame  $\tau_i$  has a configuration that changed from LO to HI values, the switch changes its criticality level to HI. When a switch detects such a change, it removes all pending frames belonging to  $\tau^{LO}$  (but keeps already received frames in  $\tau^{HI}$  sent in LO mode). All following switches in the path of  $\tau_i$  will change their criticality with the same principle (including the last node  $S_M$  in the path of  $\tau_i$ ). Due to non-preemptive transmissions, the switch can have started the transmission of one message in  $\tau^{LO}$  when a criticality switch to HI mode occurs. This result in a delay of transmission that must be taken into account in the analysis of the worst case end-to-end response time of flows in  $\tau^{HI}$  in section 4. All other switches not in the path of  $\tau_i$  will be informed of a criticality change to HI mode when receiving a specific PTP message sent by the master  $S_M$  node.

Hence, on a given node, we consider that changing the current criticality level from LO to HI is done in two situations:

- either a node (ES or switch) receives a frame  $\tau_i \in \tau^{HI}$  having characteristics that changed from  $(C_i^{LO}, T_i^{LO})$  to  $(C_i^{HI}, T_i^{HI})$ ,
- or a node receives a PTP frame from the master node that results in a criticality level change to HI mode.

The change from HI to LO mode in nodes (ES and switches) is done when receiving a specific message from the master switch to change the criticality to LO mode (in a PTP frame).

## 3 Integrating criticality management into PTP

### 3.1 Clock synchronization

Now, we want to propose the integration of criticality management into the Precision Time Protocol (PTP) frames. To do this, we first need to understand what PTP is and how it works.

IEEE 1588 PTP is a clock synchronization protocol for switched-Ethernet networks. The IEEE 1588 clock synchronization protocol used by manufacturers like CISCO or MOXA is implemented by the PTP daemon. We recall that we assume a dedicated VLAN for PTP frames.

Clock synchronization through PTP is implemented on a master-slave principle. A master clock node  $S_M$  in the network is node used to synchronize all other nodes, called slaves, with it. Indeed, with time evolving, master and slave clocks tends to desynchronize themselves. The clock synchronization operates with the exchange of frames between the master and its slaves.

First, the master sends a first synchronization message, containing a timestamp. As we are in a dedicated vln and full duplex links, there's no collision or delay generated by other flows (for communications from the master to the slaves). Once a slave gets this first synchronization messages, it answers to the master with a timestamp set with its local clock. Once the master gets this answer, it computes the correction delay for the clock of the slave. Then, the master sends a third message to the slave, containing this correction delay. Getting this delay, the slave can update its local clock accordingly.

Considering this, the delay of flows sent from ES to  $S_M$  in the network should take into account the delay induced by PTP frames sent by all End Systems.

We suppose that PTP synchronization frames are managed in a dedicated VLAN having the highest priority. The maximum delay induced by the PTP synchronization protocol on the worst case end-to-end response time of any flow (LO or HI) in the network for any flow path is denoted  $D_{Sync}$ .

### 3.2 Integrating mixed-criticality in PTP

There are two different solutions to integrate mixed-criticality management into PTP. The first one is to build dedicated frames in PTP protocol. They are in charge of sending the request of changing criticality to a given node. This solution implies to schedule messages of clock synchronization and mixed-criticality management in the PTP protocol. So, it implies a greater overhead in our network.

The second solution, which is the one we decided to choose, is to modify the clock synchronization frames in order to include criticality management in it. Then, the first PTP frame contains criticality information: if the criticality received by a node (slave switch or end system) changes, then the criticality is set to the new criticality mode (either LO or HI).

This criticality management through PTP implies greater length, so a greater WCTT in PTP frames. The PTP synchronization frames have a specific header common to all

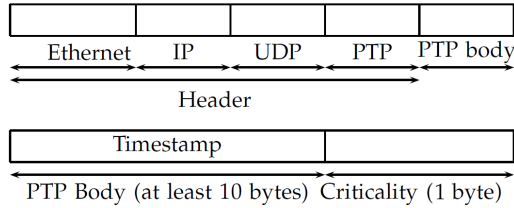


Figure 2: PTP frame format

frames (follow\_up, delay\_req and delay\_resp). This PTP frame is organized as follows in Figure 2)

The informations about current time needed to synchronize clocks is stored in the PTP body. In order to add mixed-criticality to PTP, we need to modify the structure of the PTP frame. We propose to add 1 bytes to encode the value of the criticality level (0 for LO and 1 for HI modes in our case). This one byte information leaves open the possibility to take into account more than two criticality levels.

To reduce the impact of flow set  $\tau$  on the precision of synchronization of clocks with PTP, we assume that a dedicated VLAN with the highest priority (IEEE 802.1p) is used for PTP messages.

We now consider the time needed to switch-criticality. This delay depends on time needed for the master switch to be informed of a criticality change (depending on the scheduling policy of the network) and on the PTP propagation mechanism from the master switch to all nodes.

#### 4 Switch-criticality Delay Analysis

The goal of the switch-criticality delay analysis is to find the maximum delay induced by a change of criticality level. More precisely, we identify the maximum delay from the time when a high-criticality frame enters the network running in low-criticality till the time when the whole network is aware of high-criticality. This maximum delay includes two parts. The first one is the maximum delay of a high-criticality frame which is generated at a slave node and transmitted to the master node  $S_M$ . The second part is the delay of frames sent by the master node  $S_M$  to inform the slave nodes of the high-criticality change. In order to achieve the first part of delay, we examine delay upper bounds of all the flows transmitted towards  $S_M$  by first adapting the trajectory approach introduced in [16] for FIFO scheduling policy to the context of mixed-criticality and then by extending the approach to Fixed Priority (FP) scheduling policy. For the purpose of simplicity, in this paper we do not take into account the serialization effect.

The trajectory approach considers the worst-case scenario encountered by a frame of a flow  $\tau_i$  along its path  $\mathcal{P}_i$ . A frame of flow  $\tau_i$  ( $\tau_i \in \tau^{HI}$ ) can be delayed by:

- the frames of flows in  $\tau^{HI}$  which cross the path of flow  $\tau_i$ ,
- one frame of flows in  $\tau^{LO}$  at each visited switch due to non-preemptive, and

- the frames of synchronization sent by slave nodes to the master node.

The delay upper bound of  $\tau_i$  in the context of FIFO is denoted  $D_{FIFO}(\tau_i)$  and can be split in four different parts:

- $D_{HI}(\tau_i)$ , the delay induced by all waiting high-criticality frames,
- $D_{LO}(\tau_i)$ , the delay induced by a low-criticality frame due to non-preemptive at each visited switch,
- $D_{Lat}(\tau_i)$ , the switching latency of each visited switch as well as a transition cost from one node to the next one (more details can be found in [17]),
- $D_{Sync}$ , the delay introduced by the synchronization frames. In this paper, this delay is considered as a constant delay.

We define  $\tau(S_k)$  as the set of flows which go through the switch  $S_k$ . For a frame of flow  $\tau_i$  generated at time  $t$ , we compute each part mentioned above by:

$$D_{HI}(t) = \sum_{\substack{\tau_j \in \tau^{HI} \\ \mathcal{P}_i \cap \mathcal{P}_j \neq \emptyset}} \left( 1 + \left\lfloor \frac{t + A_{i,j}}{T_j^{HI}} \right\rfloor \right) \times \max(C_j^{HI}, C_j^{LO}) \quad (1)$$

where  $t + A_{i,j}$  corresponds to a time interval in which frames from flow  $\tau_j$  are likely to delay the studied frame of  $\tau_i$  (see [17] for more details).

$$D_{LO} = \sum_{S_k \in \mathcal{P}_i} \left( \max_{\tau_j \in \tau^{LO} \cap \tau(S_k)} (C_j^{LO}) \right) \quad (2)$$

$$D_{Lat} = \sum_{S_k \in \mathcal{P}_i} \left( \max_{\tau_j \in \tau^{HI} \cap \tau(S_k)} (\max(C_j^{HI}, C_j^{LO})) \right) + (|\mathcal{P}_i| - 1) \times sl \quad (3)$$

Hence, the delay upper bound of a high-criticality frame of flow  $\tau_i$  is computed by the following equation:

$$D_{FIFO}(\tau_i) = \max_{t \geq 0} (D_{HI}(t) + D_{LO} + D_{Lat} + D_{Sync} - t) \quad (4)$$

Since the master node  $S_M$  is aware of the change of criticality level after it has received a high-criticality frame, it sends multi-cast frames to all the slave nodes in order to inform the change of criticality level. These frames are the only traffic transmitted in the direction from master node  $S_M$  to slave nodes, then the maximum delay is generated by the longest path between a slave node and the master node. We define the length of the longest path as  $|\mathcal{P}_{max}|$ , the worst transmission time of each frame sent by the master node as  $C_{Inf}$  as well as the corresponding maximum delay as  $D_{Inf}$  which is computed by:

$$D_{Inf} = |\mathcal{P}_{max}| \times (C_{Inf} + sl) \quad (5)$$

Therefore, in the context of FIFO, the maximum delay induced by the change of criticality level is obtained by:

$$D_{FIFO} = \max_{\tau_i \in \tau^{HI}} D_{FIFO}(\tau_i) + D_{Inf} \quad (6)$$

In order to extend this approach to FP scheduling policy, we consider an assumption that all the flows in  $\tau^{HI}$  have higher priorities than the flows in  $\tau^{LO}$ . In this case, the delay of a frame of  $\tau_i$  induced by other frames includes (i) the delay due to the higher-priority flows which are also in the flow set  $\tau^{HI}$ , (ii) the delay due to non-preemptive caused by lower-priority flows in the flow set  $\tau^{HI}$  and all the flows in the flow set  $\tau^{LO}$ .

Therefore, for a flow  $\tau_i$ , all the other flows can be classified into two sets: higher-priority flows and lower-priority flows no matter which criticality level the network is running. This model corresponds the flow model in [18]. Hence, the delay upper bound of a frame of flow  $\tau_i$ , denoted  $D_{FP}(\tau_i)$ , can be calculated by the trajectory approach presented in [18] integrating the constant delay  $D_{Sync}$ . The maximum delay  $D_{Inf}$  of frames sent by the master node  $S_M$  does not change since they are the only traffic in the direction master to slave.

Then in the context of mixed-criticality with FP scheduling policy, the maximum delay introduced by a change of criticality level is denoted  $D_{FP}$  and computed by:

$$D_{FP} = \max_{\tau_i \in \tau^{HI}} D_{FP}(\tau_i) + D_{Inf} \quad (7)$$

## 5 Conclusion

To conclude, we show that mixed-criticality management can be integrated to an existing protocol like PTP, just by modifying its PTP frames. This way, we can combine criticality switching and clock synchronization within switched-Ethernet networks. A switch must also be aware of the criticality of HI flows it sends in order to propose deterministic networks with mixed-criticality infrastructures. We propose a solution that does not require Vestal's constraints on the WCTT of frames. In future work, we intend to explore recent schedulability analysis techniques for Earliest Deadline First for mixed-criticality systems [19] in order to adapt them to switched Ethernet networks.

## References

- [1] S. Vestal (2007), *Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance*, in Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS), (Tucson, Arizona, USA), pp. 239–243, IEEE Computer Society.
- [2] S. K. Baruah, *Mixed criticality scheduling is highly intractable*.
- [3] S. K. Baruah, V. Bonifaci, G. D'Angelo, A. Li, Hao-han Marchetti-Spaccamela, N. Megow, and L. Stougie (2010), *Scheduling real-time mixed-criticality jobs*, Mathematical Foundations of Computer Science, vol. 6281, pp. 90–101.
- [4] S. K. Baruah, V. Bonifaci, D. Gianlorenzo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie (2011), *Mixed-criticality scheduling of sporadic task systems* in Proceedings of the 19th Annual European Symposium on Algorithms (ESA), (Saarbrücken, Germany), pp. 555–566, Springer-Verlag.
- [5] S. K. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie (2012), *The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems*, in Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), (Pisa, Italy), pp. 145–154, IEEE Computer Society Press.
- [6] S. K. Baruah, A. Burns, and R. I. Davis (2011), *Response-time analysis for mixed criticality systems*, in Proceedings of the 32nd IEEE Real-Time Systems Symposium (RTSS), (Vienna, Austria), pp. 34–43, IEEE Computer Society.
- [7] P. Ekberg and W. Yi (2012), *Bounding and shaping the demand of mixed-criticality sporadic tasks*, in Proceedings of the 24th Euromicro Conference on Real-Time Systems (ECRTS), (Pisa, Italy), pp. 135–144, IEEE Computer Society.
- [8] R. I. Davis and A. Burns (2013), *Mixed criticality on controller area network*, in Proceedings of the 25th Euromicro Conference on Real-time Systems (ECRTS), (Paris, France), pp. 125–134, IEEE Computer Society.
- [9] D. Goswami, M. Lukasiewicz, R. Schneider, and S. Chakraborty (2012), *Time-triggered implementations of mixed-criticality automotive software*, in Proceedings of the Conference on Design, Automation and Test in Europe (DATE), (Dresden, Germany), pp. 1227–1232, IEEE Computer Society.
- [10] V. Sciandra, P. Courbin, and L. George (2012), *Application of mixed-criticality scheduling model to intelligent transportation systems architectures*, in ACM SIGBED Review - Special Issue on the Work-in-Progress (WiP) session of the 33rd IEEE Real-Time Systems Symposium (RTSS), vol. 10, (San Juan, Puerto Rico), p. 22, ACM Press.
- [11] A. Addisu, L. George, V. Sciandra, and M. Agueh (2013), *Mixed criticality scheduling applied to jpeg2000 video streaming over wireless multimedia sensor networks*, in Proceedings of the 1st International Workshop on Mixed Criticality Systems (WMCS), (Vancouver, Canada), pp. 55–60.
- [12] R. Obermaisser (2013), *Distributed REal-time Architecture for Mixed criticality Systems (DREAMS)*, url: <http://www.uni-siegen.de/dreams/>.
- [13] S. Medlej, S. Martin, and J.-M. Cottin (2012), *Identifying source of pessimism in the trajectory approach with fifo scheduling*, in Proceedings of Embedded Real-Time Software and Systems (ERTS<sup>2</sup>), (Toulouse, France).

- [14] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard (2013), *Optimistic problems in the trajectory approach in fifo context*, in Proceedings of the 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), (Cagliari, Italy), IEEE Computer Society.
- [15] X. Li, O. Cros, and L. George (2014), *The trajectory approach for afdx fifo networks revisited and corrected*, in Proceedings of the 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), (Chongqing, China), IEEE Computer Society.
- [16] S. Martin (2004), *Maîtrise de la dimension temporelle de la qualité de service dans les réseaux*. PhD thesis, Université Paris XII Val de Marne.
- [17] S. Martin and P. Minet, “Schedulability analysis of flows scheduled with fifo: application to the expedited forwarding class,” in *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, (Rhodes Island, Greece), p. 8, IEEE Computer Society, April 2006.
- [18] S. Martin and P. Minet (2006), *Worst case end-to-end response times of flows scheduled with FP/FIFO*, in Proceeding of International Conference on Networking (ICN), (Mauritius), pp. 54–60.
- [19] P. Ekberg and W. Yi (2014), *Bounding and shaping the demand of generalized mixed-criticality sporadic task systems*, in *Real-Time Systems*, vol. 50, pp. 48–86.

# Mixed Criticality in Railway Systems: A Case Study on Signalling Application

*Albert Cohen, Dumitru Potop-Butucaru*

*INRIA, France; email: {albert.cohen, dimitru.potop\_butucaru}@inria.fr*

*Valentin Perrelle, Zhen Zhang*

*Technological Research Institute SystemX; email: {valentin.perrelle, zhen.zang}@irt-systemx.fr*

*Elie Soubiran*

*Technological Research Institute SystemX, Alstom Transport; email: elie.soubiran@irt-systemx.fr;  
elie.soubiran@transport.alstom.com*

## 1 Introduction

Since the early 2000's a growing number of new automated metro projects makes use of a standardized railway signalling system called Communication Based Train Control (CBTC) (IEEE 1474) [1]. Previously to CBTC, conventional signalling train control systems were relying almost exclusively on track circuits, wayside signals and operating procedures to ensure train protection and operation. In order to ensure better operational performance (e.g. effective utilization of the transit infrastructure), CBTC systems rest on three pillars: "Automatic train control (ATC) based on high-resolution train location determination, independent of track circuits"; "high-capacity and bidirectional train-to-wayside data communications"; and "train-borne and wayside computing units that execute vital functions". Functions are classified within three families that are: Automatic Train Protection (ATP), Automatic Train Operation (ATO) and Automatic Train Supervision (ATS). The level of criticality differs from a family to another and without loss of generality, one can state that ATP functions are mostly safety critical functions (SIL4 regarding CENELEC 50126), whereas ATO and ATS gather functions of low criticality (from SIL0 to SIL2). As a matter of fact, CBTC systems are in essence Mixed-critical systems. Furthermore, the mainstream evolution of those systems tends toward more functional integration on more powerful computing units. ATP and ATO functions that were traditionally distributed on different computing units (both on wayside and train-borne) tends now to be deployed on the same computing units and thus sharing resources.

FSF<sup>1</sup> is an IRT SystemX project positioned on two topics, the first one is about the conception of signalling applications (typically ATO/ATP application) that contain both critical and non-critical parts and the second one is on execution platforms that execute those applications while offering high guarantee of safety and availability. Industrial expectations around the execution platform include the use of multi-core

<sup>1</sup>FSF is a French project name acronym standing for "safe and reliable embedded systems"

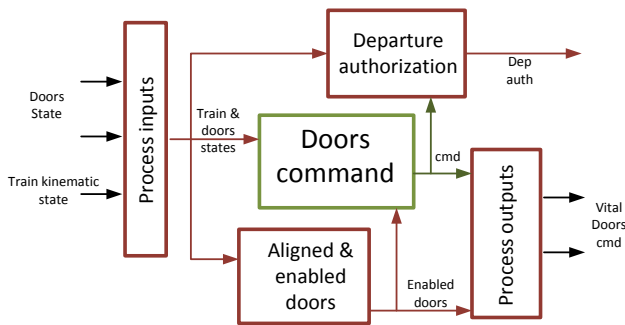
COTS, the use of modern RTOS that offer spatial and temporal isolation, the use of safety and availability architectural patterns (e.g. voting and redundancy), and the whole being finally hidden behind a "system abstraction layer".<sup>2</sup> On top of this platform, a tooling framework is prototyped and allows one to develop, verify and deploy component based applications where components may arbitrary contains both vital (SIL4) and non-vital (SIL0) code. The project has started in May 2013, the aim of this communication is to propose a first return of experience and a positioning on how mixed critical issues will be addressed in FSF.

Alstom Transport has defined an applicative case study that, while being limited to one single ATC function, is representative of the complexity in term of vital/non-vital code interweaving, operational performance and availability constraints. The system function is called "Passenger Exchange" (PE). This function takes control on the train when this one is safely docked at a station; it organizes the exchange of passengers (train and station doors opening/closing) while protecting them from any untimely train movement or non-aligned doors opening and finally gives the departure authorization when all safety conditions are met. The functional specification is made of more than 300 requirements (natural language + SysML), and the functional structure is made of about twenty sub functions.

PE is designed as a system component containing both vital and non-vital parts. At this level a component is roughly a packaging unit that exposes to the exterior world a set of ports (in or out) and that is characterized by a set of behaviours that depend on the operational environment. This component is then broken down into a set of atomic software components which are this time exclusively vital or non-vital. An important remark is that there are no restrictive design constraints on data dependency between the vital and non-vital atomic components.

To illustrate this fact, let us consider a simplified example from the case study, depicted in Fig. 1. The vital components,

<sup>2</sup>The execution platform is not yet prototyped and will not be described in this communication.



**Figure 1: Simplified view of the component breakdown in PE.**

in red, are in charge of controlling operations. This can be summarized by computing which doors are safe to open (e.g. because they are not aligned) and by preventing train departure if safety conditions are not met (e.g. the doors are open or opening). The non-vital components are in charge of operating doors with respect to a mission protocol and a time table. In this example we can identify two occurrences of vital to non-vital communications. First, in order to ensure that doors commands (non-vital) do not lead to an accident, they must be checked against the enabled set of doors (vital). This is the role of the “Process output” atomic component. Second, the departure authorization (vital) must be computed regarding door commands to ensure that no opening commands will be executed after the authorization has been given.

This is a simplified example. Such communication patterns are quite common in the complete case study.

## 2 Synchronous approaches

### 2.1 Synchronous languages

Data-flow synchronous languages, such as LUSTRE [2] or SIGNAL [3] have been designed in the 80’s for program real-time safety critical embedded systems. Since then, they have been widely used in industrial applications [4]. These languages emphasize a correct-by-construction approach, ensuring bounded memory and execution time. Moreover, they are praised for their predictable behaviour and formally defined semantics.

Recently, in 2012, the problem of scheduling multi-rate, mixed-critical synchronous programs have been addressed, at first for uni-processor [5] then for multi-processors [6]. Outside the scope of mixed-criticality there were also several attempts to distribute synchronous data-flow languages [7, 8, 9]. Still in 2012, work have been done to develop these languages to target multi-core platforms through the programming of parallelism [10]. This work introduces futures in LUSTRE-like languages giving the guarantee that the sequential semantics is preserved.

### 2.2 Automatic allocation, partitioning, and scheduling

Due to their use in the avionics industry, synchronous languages have been considered early on as an input formalism for the automatic or semi-automatic synthesis of real-time implementations. Most significant in this direction are

previous results by Sorel *et al.* [11] on the AAA/SynDEX methodology and tool for distributed, but not time-triggered, real-time implementation of multi-periodic synchronous specifications, previous work by Caspi *et al.* on the use of LUSTRE/SCADE in the real-time implementation of Simulink over multi-processor platforms based on the time-triggered partitioned bus TTA [12], and previous work by Forget *et al.* [13] on the specification and implementation of multi-periodic applications over a time-triggered platform using the Prelude language.

But none of these approaches allow us to take into account all the characteristics of our case study in order to allow automatic mapping. In particular, none of them has support for ensuring the time and space separation between application parts with different *criticalities*.

This is why we considered in this project a new tool, named LOPHT [14, 15], which allows the automatic mapping of applications onto platforms following the ARINC 653 time and space partitioning mechanisms. The LOPHT tool takes as input deterministic functional specifications provided by means of synchronous data-flow models with multiple modes and multiple relative periods. (Cf. the LOPHT part of Fig. 2) These specifications are extended to include a real-time characterization defining task periods, release dates, and deadlines. Task deadlines can be longer than the period to allow a faithful representation of complex end-to-end flow requirements. The specifications are also extended with allocation constraints and partitioning information meant to represent the criticality of the various tasks, as well as information on the preemptability of the various tasks. Starting from such specifications, the LOPHT tool performs a fully automatic allocation and off-line scheduling onto partitioned time-triggered architectures. Allocation of time slots/windows to partitions can be fully or partially provided, or synthesized by LOPHT. The mapping algorithms of LOPHT take into account the communication costs. The off-line mapping algorithms of LOPHT use advanced mapping techniques such as software pipelining and pre-computed preemption to improve schedulability and minimize the number of context switches.

## 3 Case study

The PE case study has been implemented and a first demonstrator has been produced. The challenge for this first demonstrator was to propose a framework for on the one hand the design and implementation of components and on the other hand the design of signalling application its partitioning and scheduling.

**Choice of software modelling language.** We chose to use the language HEPTAGON, very similar to LUSTRE and featuring novel constructions and novel optimizations. Two criteria have influenced the choice of the language. First, the functional specification defined at system level and allocated to software components have been written in a reactive and mostly equational way. It was thus very natural to implement it in a synchronous data-flow language. Second, the normative referential (CENELEC 50128) recommends the use of formal methods for the development of vital software while making

no restrictive assumption on the language used for the non vital part. Synchronous languages are a good trade-off since they enable the use of formal methods (for instance model checking or abstract interpretation) while providing a sufficient power of expression to implement non-vital components. Finally, having a single language to develop both vital and non-vital components allows not only the early simulation of functional behaviour without integration effort but also the rationalization of competence in the software development team.

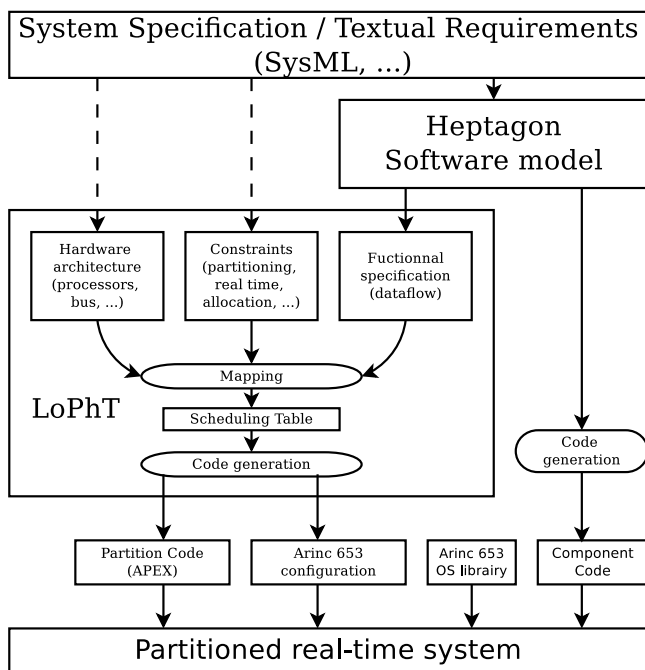


Figure 2: The global flow of the use case.

**Technical realization.** We developed the Passenger Exchange sub-components following a five step process depicted in Fig. 2:

1. In a SysML environment, we produced a component design that realizes the Passenger Exchange function. System requirements are traced and refined to define atomic components that correspond to software components and that are either vital or non-vital.
2. We mapped every atomic component to a HEPTAGON node realizing its functional behaviour.
3. Depending on the SIL of the component, verification activities have been conducted, but these are outside the scope of this communication.
4. We built a small signalling application composing multiple components into a realistic full-system scenario. These components include the PE itself, train/station interfaces, and a simulation of other system functions (such as train driving and passenger behavior). In HEPTAGON, the application is an assembly of nodes. At this stage, a first executable code is produced to simulate the application behaviour, however no insurance is given on spatial isolation.

5. In LOPHT, the atomic components are allocated to three partitions, which are “P0: vital”, “P1: non-vital” and “P2: environment”. Meanwhile the execution durations are given. Five windows are created. The scheduling result, presented in the Fig. 3, is consistent with the block diagram presented in Fig. 1. Using LOPHT, ARINC 653 dependent code is generated and linked to the component code generated by HEPTAGON. The resulting application is executed on POK OS [16].

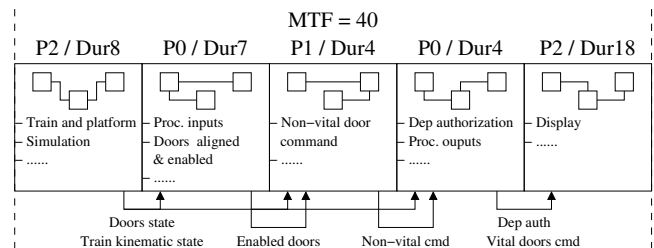


Figure 3: The partitionial scheduling result of LOPHT.

## 4 Conclusion

We presented the work conducted in the FSF project to handle mixed criticality. We used a synchronous design framework to implement a simplified signalling application and to deploy it on a partitioned OS.

We are continuously working towards a better integration of the tools composing the framework.

In the passenger exchange use case, mixed criticality resides at the application level, or even at function level, rather than the system level. On the other hand, the approach proposed in IMA and ARINC meets the needs of a system integrator. The main constraint highlighted by this case study is that there may be, even within a single system function, many communications between its vital and non-vital subcomponents. When generalized to the whole set of system functions, this pattern induces a large number of communications between the vital and non-vital parts. Furthermore, if we want to preserve the synchronous semantics (e.g. no additional delay) the number of windows may explode. The overall cost of communications and context-switch become prohibitive for systems global performance. Executing mixed-critical signalling applications on the same platform remains a challenging problem considering the state of the art in real-time operating systems.

Finally, the vital/non-vital dichotomy traditionally used in signalling application proved to be insufficient with respect to the operational availability of the system. It would be more appropriate to consider at least three levels, safety-critical, mission-critical, and non-critical, and to exploit this information in the partitioning and scheduling.

## Acknowledgment

This research work has been carried out under the leadership of the Technological Research Institute SystemX, with partial support from the French Program “Investissements d’Avenir”.



## References

- [1] UITP, “World Atlas Report,” UITP - Observatory of Automated Metros, Survey, 2013. [Online]. Available: <http://metroautomation.org/wp-content/uploads/2013/09/Annual-World-Report-2013.pdf>
- [2] P. Caspi, D. Pilaud, N. Halbwachs, and J. A. Plaice (1987), *Lustre: A declarative language for real-time programming*, in Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, ser. POPL ’87. New York, NY, USA: ACM, pp. 178–188. [Online]. Available: <http://doi.acm.org/10.1145/41625.41641>
- [3] A. Benveniste, P. Le Guernic, and C. Jacquemot (1991), *Synchronous programming with events and relations: The signal language and its semantics*, Sci. Comput. Program., vol. 16, no. 2, pp. 103–149. [Online]. Available: [http://dx.doi.org/10.1016/0167-6423\(91\)90001-E](http://dx.doi.org/10.1016/0167-6423(91)90001-E)
- [4] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. L. Guernic, Robert, and D. Simone (2003), *The synchronous languages 12 years later*, in Proceedings of The IEEE, pp. 64–83.
- [5] S. Baruah (2012), *Semantics-preserving implementation of multirate mixed-criticality synchronous programs*, in Proceedings of the 20th International Conference on Real-Time and Network Systems, ser. RTNS ’12. New York, NY, USA: ACM, pp. 11–19. [Online]. Available: <http://doi.acm.org/10.1145/2392987.2392989>
- [6] E. Yip, M. M. Y. Kuo, P. S. Roop, and D. Broman (2014), *Relaxing the synchronous approach for mixed-criticality systems*, in Proceedings of the Work in Progress Session of the 20th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS’14 WiP). IEEE, 2014.
- [7] P. Aubry and P. Le Guernic (1996), *On the desynchronization of synchronous applications*, in 11th International Conference on Systems Engineering, ICSE, vol. 96. Citeseer.
- [8] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, and P. Niebert (2003), *From simulink to scade/lustre to tta: A layered approach for distributed embedded applications*, in Proceedings of the 2003 ACM SIGPLAN Conference on Language, Compiler, and Tool for Embedded Systems, ser. LCTES ’03. New York, NY, USA: ACM, pp. 153–162. [Online]. Available: <http://doi.acm.org/10.1145/780732.780754>
- [9] G. Delaval, A. Girault, and M. Pouzet (2008), *A type system for the automatic distribution of higher-order synchronous dataflow programs*, in SIGPLAN Not., vol. 43, no. 7, pp. 101–110. [Online]. Available: <http://doi.acm.org/10.1145/1379023.1375672>
- [10] A. Cohen, L. Gérard, and M. Pouzet (2012), *Programming parallelism with futures in lustre*, in Proceedings of the Tenth ACM International Conference on Embedded Software, ser. EMSOFT ’12. New York, NY, USA: ACM, pp. 197–206. [Online]. Available: <http://doi.acm.org/10.1145/2380356.2380394>
- [11] M. Marouf, L. George, and Y. Sorel (2012), *Schedulability analysis for a combination of non-preemptive strict periodic tasks and preemptive sporadic tasks*, in Proceedings ETFA’12, Kraków, Poland.
- [12] P. Caspi, A. Curic, A. Magnan, C. Sofronis, S. Tripakis, and P. Niebert (2003), *From Simulink to SCADE/Lustre to TTA: a layered approach for distributed embedded applications*, in Proceedings LCTES, San Diego, CA, USA.
- [13] C. Pagetti, J. Forget, F. Boniol, M. Cordovilla, and D. Lesens (2011), *Multi-task implementation of multi-periodic synchronous programs*, in Discrete Event Dynamic Systems, vol. 21, no. 3, pp. 307–338.
- [14] T. Carle, D. Potop-Butucaru, Y. Sorel, and D. Lesens (2012), “From dataflow specification to multiprocessor partitioned time-triggered real-time implementation,” INRIA, Rapport de recherche RR-8109, Oct. 2012. [Online]. Available: <http://hal.inria.fr/hal-00742908>
- [15] T. Carle and D. Potop-Butucaru (2014), *Predicate-aware, makespan-preserving software pipelining of scheduling tables*, in ACM Trans. Archit. Code Optim., vol. 11, no. 1, pp. 12:1–12:26.
- [16] J. Delange, L. Pautet, and P. Feiler (2009), *Validating safety and security requirements for partitioned architectures*, in Proceedings of the 14th Ada-Europe International Conference on Reliable Software Technologies, ser. Ada-Europe ’09. Berlin, Heidelberg: Springer-Verlag, pp. 30–43.

# 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](mailto:Dirk.Craeynest@cs.kuleuven.be)  
 URL: [www.cs.kuleuven.be/~dirk/ada-belgium](http://www.cs.kuleuven.be/~dirk/ada-belgium)

## Ada in Denmark

attn. Jørgen Bundgaard  
 Email: [Info@Ada-DK.org](mailto:Info@Ada-DK.org)  
 URL: [Ada-DK.org](http://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](mailto:Hubert.Keller@kit.edu)  
 URL: [ada-deutschland.de](http://ada-deutschland.de)

## Ada-France

attn: J-P Rosen  
 115, avenue du Maine  
 75014 Paris  
 France  
 URL: [www.ada-france.org](http://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](mailto:ssaez@disca.upv.es)  
 URL: [www.adaspain.org](http://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](mailto:rei@ada-sweden.org)  
 URL: [www.ada-sweden.org](http://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](mailto:president@ada-switzerland.ch)  
 URL: [www.ada-switzerland.ch](http://www.ada-switzerland.ch)