ADA USER JOURNAL

Volume 40
Number 4
December 2019

Contents

Page

Editorial Policy for *Ada User Journal* 188

Editorial 189

Quarterly News Digest 190

Conference Calendar 197

Forthcoming Events 205

Ada-Europe 2019 Industrial Presentations

R. Bramberger, H. Martin, B. Gallina, C. Schmittner
"Co-engineering of Safety and Security Life Cycles for Engineering of Automotive Systems" 210

Proceedings of the "Workshop on Challenges and New Approaches for Dependable and Cyber-Physical Systems Engineering" of Ada-Europe 2019

J. Górski
"Using Evidence-Based Arguments to Support Dependability Assurance – Experiences and Challenges" 219

T. Naks, M. A. Aiello, S. T. Taft
"Using SPARK to Ensure System to Software Integrity: A Case Study" 226

J. J Valls, M. García-Gordillo, S. Sáez
"Scenario-Based Validation & Verification, the ENABLE-S3 Approach" 230

R. Tomar, Sarishma
"Maintaining Trust in VANETs Using Blockchain" 236

Article

P. Neto, J. Tojal, J. Veríssimo, S. Melo de Sousa
"Towards a Formally Verified Space Mission Software Using SPARK" 243

Ada-Europe Associate Members (National Ada Organizations) 248

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*.

The Journal publishes the following types of material:

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

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

Original Papers

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

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

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

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

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

Proceedings and Special Issues

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

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

News and Product Announcements

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

Reprinted Articles

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

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

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

Editorial

As some may already have noticed, the Quarterly News Digest section that we offer to our readers in every issue, which used to become publicly available after one year (with the respective issue), is now being made publicly available on the AUJ archive, as a separate pdf document, 6 months after the issue date. Given the informative character of this News Digest and the fact that the selection of news it provides comes from public sources, we considered that by shortening its embargo period we would increase its relevance and usefulness for a larger audience, while still providing a differentiated service to our subscribers. In addition, each newly publicly released Quarterly News Digest is now also being announced and made available by the AdaIC, through a link posted on the adaic.org website. We hope that this change and the collaboration with the AdaIC will be well received by the Ada community.

As for the contents of this issue, we continue the publication of contributions related to the 24th International Conference on Reliable Software Technologies (Ada-Europe 2019) that took place in Warsaw last June, and we also include an article that was directly submitted to the journal.

To conclude the proceedings of the industrial track of the 2019 Ada-Europe, the reader will find a paper that addresses how to deal with safety and security concerns and their mutual implications when engineering safety-critical automotive systems. The paper, co-authored by researchers from the Virtual Vehicle Research Center and from the Austrian Institute of Technology (both in Austria), as well as from the Mälardalen University in Sweden, proposes a safety/security co-engineering process that fills a gap in currently existing standards, which do not define any such process.

Then we continue the publication of the Workshop on Challenges and New Approaches for Dependable and Cyber-Physical Systems Engineering (DeCPS 2019) proceedings, with a set of four papers addressing several interesting topics. The first one, by Janusz Górski from the Gdańsk University of Technology in Poland, addresses the problem of evidence-based arguments and their applications, referring to challenges that were faced when developing and deploying concrete solutions to that problem. Then, from AdaCore affiliated authors, the second paper describes work in progress on a workflow that builds on SysML to Simulink and Simulink to SPARK translations and supports consistent property-preservation proofs from early stages of system requirement specifications down. The third paper is authored by researchers from the Instituto Tecnológico de Informática in Spain, and describes work done in the context of the ENABLE-S3 project concerning a solution developed for validation and verification processes and applied to a Reconfigurable Video Processor (RVP) for space missions. Last but not the least, the fourth paper, by authors from two universities in Dehradun, India, addresses the important security issue on how to maintain trust in ad-hoc vehicular networks, describing an approach that relies on blockchains.

Finally, the issue concludes with an article by Portuguese authors, from Critical Software S.A. and from the University of Beira Interior, which introduces the implementation, verification and validation of the ExoMars Trace Gas Orbiter (TGO) central software that was implemented by Critical Software, S.A. in cooperation with Thales Alenia Space for the European Space Agency (ESA) Mars exploration mission.

The Quarterly News Digest and Calendar sections are included as usual, prepared respectively by Alejandro R. Mosteo and Dirk Craeynest, their editors.

Antonio Casimiro
Lisboa
December 2019
Email: AUJ_Editor@Ada-Europe.org

Quarterly News Digest

Alejandro R. Mosteo

Centro Universitario de la Defensa de Zaragoza, 50090, Zaragoza, Spain; Instituto de Investigación en Ingeniería de Aragón, Mariano Esquillor s/n, 50018, Zaragoza, Spain; email: amosteo@unizar.es

Contents

Ada-related Events 190
Ada-related Resources 191
Ada-related Tools 192
Ada and Operating Systems 194
Ada and other Languages 194
Ada Practice 195

Ada-related Events

25th Ada-Europe Int'l Conf. on Reliable Software Technologies

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Subject: CfC 25th Ada-Europe Conf. on Reliable Software Technologies
Date: Fri, 25 Oct 2019 05:29:17 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada,comp.lang.misc

Call for Contributions

25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2020)

8-12 June 2020, Santander, Spain

www.ada-europe.org/conference2020

Organized by University of Cantabria and Ada-Europe

#AdaEurope #AEiC2020 #AdaProgramming

General Information

The 25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2020 aka Ada-Europe 2020) will take place in Santander, Spain, in the week of 8-12 June. The conference schedule includes a technical program and vendor exhibition, and parallel tutorials and workshops.

The 2020 edition of the conference continues the major revamp in the registration fees introduced in 2019, redesigned to extend participation from industry and academia, and to reward contributors, especially but not solely, students and post-doc researchers.

[More information in the forthcoming events section of the Journal. --arm]

10th Ada Developer Room at FOSDEM 2020

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Subject: CfP - Ada Developer Room at FOSDEM 2020, Brussels, Belgium
Date: Sat, 26 Oct 2019 06:43:59 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada

[NB: The call for contributions has ended but it is kept for reference. The program is given in the Forthcoming Events section of this AUJ issue --arm]

Call for Presentations

10th Ada Developer Room at FOSDEM 2020

Saturday 1 February 2020, Brussels, Belgium

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem.html

Organized in cooperation with Ada-Europe

Ada-Belgium [1] is pleased to announce there will be a one-day Ada Developer Room on Saturday 1 February 2020 at FOSDEM 2020 in Brussels, Belgium. Our 10th Ada DevRoom is once more organized in cooperation with Ada-Europe [2].

General Information

FOSDEM [3], the Free and Open source Software Developers' European Meeting, is a free and non-commercial two-day weekend event organized early each year in Brussels, Belgium. It is highly developer-oriented and brings together 8000+ participants from all over the world.

No registration is necessary.

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

Ada Programming Language and Technology

Awareness of safety and security issues in software systems is ever increasing. Multi-core platforms are now abundant. These are some of the reasons that the Ada programming language and technology attracts more and more attention, among others due to Ada's support for programming by contract and for multi-core targets. The latest Ada language definition was updated early 2016. Work on new features is ongoing, such as improved support for fine-grained parallelism, and will result in a new Ada standard scheduled for 2021. Ada-related technology such as SPARK provides a solution for the safety and security aspects stated above.

More and more tools are available, many are open source, including for small and recent platforms. Interest in Ada keeps further increasing, also in the open source community, and many exciting projects have been started.

Ada Developer Room

FOSDEM is an ideal fit for an Ada Developer Room. On the one hand, it gives the general open source community an opportunity to see what is happening in the Ada community and how Ada technology can help to produce reliable and efficient open source software. On the other hand, it gives open source Ada projects an opportunity to present themselves, get feedback and ideas, and attract participants to their project and collaboration between projects.

At previous FOSDEM events, the Ada-Belgium non-profit organization organized successful Ada Developer Rooms, offering a full day program in 2006 [4], a two-day program in 2009 [5], and full day programs in 2012-2016 [6-10], and in 2018-2019 [12]. An important goal is to present exciting Ada technology and projects also to people outside the traditional Ada community.

Our proposal for another dedicated Ada DevRoom was accepted, and now work continues to prepare the detailed program. We most probably will have a total of 8.5 schedulable hours between 10:30 and 19:00 in one of the rooms which accommodate from 59 to 85 participants.

More information will be posted on the dedicated web-page on the Ada-Belgium site [13], and final announcements will of course also be sent to various forums, lists and newsgroups.

Call for Presentations

We would like to schedule technical presentations, tutorials, demos, live performances, project status reports, discussions, etc, in the Ada Developer Room.

Ada-Belgium calls on you to:

- inform us at ada-belgium-board@cs.kuleuven.be about specific presentations you would like to hear in this Ada DevRoom;
- for bonus points, subscribe to the Ada-FOSDEM mailing list [14] to discuss and help organize the details;
- for more bonus points, be a speaker: the Ada-FOSDEM mailing list is the place to be!

Do you have a talk you want to give?

Do you have a project you would like to present?

Would you like to get more people involved with your project?

We're inviting proposals that are related to Ada software development, and include a technical oriented discussion.

You're not limited to slide presentations, of course.

Be creative. Propose something fun to share with people so they might feel some of your enthusiasm for Ada!

Speaking slots are 15 or 45 minutes, plus 5 minutes for Q&A. Depending on interest, we might also have a session with lightning presentations (e.g. 5 minutes each), and/or an informal discussion session.

Note that all talks will be streamed live (audio+video) and recorded, for remote as well as later viewing of talks, and so that people can watch streams in the hallways when rooms are full. By submitting a proposal, you agree to being recorded and streamed, and agree the content of your talk will be published under the same license as all FOSDEM content, a Creative Commons (CC-BY) license.

Submission Guidelines

Subscribe to the Ada-FOSDEM mailing list [14], and submit your proposal there. If needed, feel free to contact us at ada-belgium-board@cs.kuleuven.be.

Please include:

- your name, affiliation, contact info;
- the title of your talk (be descriptive and creative);
- a short descriptive and attractive abstract;
- potentially pointers to more information;
- a short bio and photo.

See programs of previous Ada DevRooms (URLs below) for presentation examples, as well as for the kind of info we need.

We'd like to put together a draft schedule by end of November. So, please act ASAP, the sooner the better, but definitely by November 25, 2019.

We look forward to lots of feedback and proposals!

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

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

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

[3] https://fosdem.org

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

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

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

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

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

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

[10] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/16/160130-fosdem.html

[11] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/18/180203-fosdem.html

[12] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/19/190202-fosdem.html

[13] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem.html

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

Ada-related Resources

[Delta counts are from Aug 7th to Dec 2nd. --arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: Mon, 2 Dic 2019
To: Ada User Journal readership

Ada groups on various social media:

- LinkedIn: 2_896 (+48) members	[1]
- Reddit: 2_420 (+113) members	[2]
- StackOverflow: 1746 (+61) questions	[3]
- Freenode: 85 (+9) users	[4]
- Gitter: 44 (+2) people	[5]
- Telegram: 50 (+5) users	[6]
- Twitter: 73 (+41) tweeters	[7]
89 (+53) unique tweets	[7]

[1] https://www.linkedin.com/groups/114211/

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

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

[4] #Ada on irc.freenode.net

[5] https://gitter.im/ada-lang

[6] https://t.me/ada_lang

[7] http://bit.ly/adalang-twitter

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: Aug 23, 2019
To: Ada User Journal readership

GitHub: 576 (+3) developers	[1]
Rosetta Code: 707 (+41) examples	[2]
38 (+2) developers	[3]
Sourceforge: 271 (+1) projects	[4]
Open Hub: 211 (+2) projects	[5]
Bitbucket: 88 (+1) repositories	[6]
Codelabs: 49 (+2) repositories	[7]
AdaForge: 8 (=) repositories	[8]

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

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

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

[4] https://sourceforge.net/directory/language:ada/

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

[6] https://bitbucket.org/repo/all?name=ada&language=ada

[7] https://git.codelabs.ch/?a=project_index

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

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: Thu May 23 2019
To: Ada User Journal readership

[NB: positive ranking changes means to go down in the ranking. --arm]

- TIOBE Index: 36 (-1) 0.296% (=) [1]
- IEEE Spectrum (general): 43 (+1) Score: 24.8 [2]
- IEEE Spectrum (embedded): 13 (=) Score: 24.8 [2]

[1] https://www.tiobe.com/tiobe-index/

[2] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019

Exercism

From: AdaMagica <christ-usch.grein@t-online.de>
Subject: Exercism
Date: Wed, 2 Oct 2019 04:43:17 -0700
Newsgroups: comp.lang.ada

Accidentally, I came across Exercism (not Exorcism):

https://exercism.io/

Would it be appropriate to add Ada to the list of languages?

[Exercism seems to be a learning and mentorship website. From their website: “Level up your programming skills with 3,094 exercises across 52 languages, and insightful discussion with our dedicated team of welcoming mentors. Exercism is 100% free forever.” --arm]

From: Lucretia <laguest9000@googlemail.com>
Date: Wed, 2 Oct 2019 05:22:09 -0700

> Would it be appropriate to add Ada to the list of languages?

Yes. From the FAQ:

How do new language tracks get added to the site?

A new language track gets created when a member of the community takes the lead on it and becomes a maintainer of the track. If you'd like to get involved in helping set one up, there are instructions here.

From: Lucretia <laguest9000@googlemail.com>
Date: Wed, 2 Oct 2019 05:23:22 -0700

Looks like they have it started: https://github.com/exercism/ada

Ada-related Tools

Gnu Emacs Ada Mode 6.2.1

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: Gnu Emacs Ada mode 6.2.1 released.
Date: Mon, 19 Aug 2019 04:46:27 -0700
Newsgroups: comp.lang.ada

Gnu Emacs Ada mode 6.2.1 is now available in GNU ELPA. This is a minor feature and bug fix release.

The elisp parser is deleted.

Ada-mode now supports some simple refactor operations; convert between Object.Method and Prefix.Method (Object) syntax.

ada-mode now provides a project.el backend. `project-find-file' does file name completion on files in the current project, using the `uniquify-files' completion style. To use this backend with an existing Ada mode project file:

(setq ada-project-current (make-ada-project :ada-prj-file <exising-file.prj>))

(add-to-list 'project-find-functions #'ada-project-current)

Error correction is faster.

See the NEWS files in ~/.emacs.d/elpa/ada-mode-6.2.1 and wisi-2.2.1, or at http://www.nongnu.org/ada-mode/, for more details.

The process parser requires a manual compile step, after the normal list-packages installation:

```
cd ~/.emacs.d/elpa/ada-mode-6.2.1
./build.sh
```

This requires AdaCore gnatcoll packages which you may not have installed; see ada-mode.info Installation for help in installing them.

GNAT for LLVM

From: Lucretia <laguest9000@googlemail.com>
Subject: Well, they kept that quiet!
Date: Mon, 30 Sep 2019 06:09:02 -0700
Newsgroups: comp.lang.ada

It's not on their blog yet, but they sneaked in this onto GitHub:

https://github.com/AdaCore/gnat-llvm

Will require the gprbuild from GitHub by the looks of it as well, unless you compilers.xml has support already inside for GNAT_LLVM as of June:

https://github.com/AdaCore/gprbuild/search?q=llvm&unscoped_q=llvm

This shall be interesting…

From: Simon Wright <simon@pushface.org>
Date: Mon, 30 Sep 2019 17:34:10 +0100

> This shall be interesting...

Indeed!

GNAT_LLVM is in gprbuild in the macOS 9.1.0 build I put on Sourceforge, as well as in the macOS CE 2019.

From: Arnaud Charlet
Subject: Combining GNAT with LLVM
Date: Tue, 01 Oct 2019
URL: https://blog.adacore.com/combining-gnat-with-llvm

[What follows is an excerpt from the post at AdaCore’s Blog. --arm]

At AdaCore labs, we have been working for some time now on combining the GNAT Ada front-end with a different code generator than GCC. [...]

This time, we're looking at another general purpose code generator, called LLVM, in order to expand the outreach of Ada to the LLVM ecosystem (be it the compiler itself or other components such as static analysis tools).

This work-in-progress research project is called "GNAT LLVM" and is meant to show the feasibility of generating LLVM bytecode for Ada and to open the LLVM ecosystem to Ada, including tools such as KLEE, that we are also planning to work with and add Ada support for. Note that we are not planning on replacing any existing GNAT port based on GCC, so this project goes in addition rather than in replacement.

[...]

Simple Components for Ada 4.42 with JSON (RFC 7158)

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple components for Ada v.4.42
Date: Mon, 16 Sep 2019 08:48:35 +0200
Newsgroups: comp.lang.ada

The new version provides an implementation of JSON (RFC 7158)

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

The JSON implementation supports both parsing and output of JSON objects. The parser allocates parts of the JSON object in a user-provided storage pool that can be an arena stack allowing both performance and safety by limiting the overall size of the object.

From: onox <denkpadje@gmail.com>
Date: Thu, 3 Oct 2019 16:53:44 -0700

> Hi Dmitry,

> I would be curious to know the differences compared to what GNATCOLL.JSON provides, if you know? It would be nice if the Ada world did not have too many packages competing here.

Uh oh, I've written a JSON parser ([0]) too! Sorry. It's quite fast, only about 1K SLOC, supports Ada 2012's iterator and indexing syntax, Apache 2.0 license, but it does not handle UTF-8 yet (patches welcome though)

> Various things which I find limiting in GNATCOLL.JSON: performance is not really good because there are a lot of memory allocations

I did some benchmarking of some JSON parsers using a 110 M large .json file from [1]:

- Parsers.JSON choked on it (I used Parsers.Multiline_Source.Stream_IO)
- GNATCOLL.JSON needs about ~ 17 seconds
- json-ada needs about ~ 3 seconds (should be ~ 1 second to be competitive with other languages, patches/advice welcome)

[1] https://github.com/onox/json-ada

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 4 Oct 2019 14:15:08 +0200

> Parsers.JSON choked on it

I see. The reason why this is not parsed is the stack size. The source contains a JSON array of 1000000 elements. In order to be parsed there should be possible to pass an array of 1M arguments. If I correctly remember it could be done like this:

```
gprbuild ... -largs -Wl,
--stack=0x8000000
```

How relevant such samples are is another question.

XNAdaLib 2019

From: Blady <p.p11@orange.fr>
Subject: [ANN] XNAdaLib 2019 binaries for macOS High Sierra including GTKAda and more.
Date: Sun, 13 Oct 2019 21:17:59 +0200
Newsgroups: comp.lang.ada

This is XNAdaLib 2019 built on macOS 10.13 High Sierra for Native Quartz with GNAT Community 2019 including:

- GTKAda 19.0w mid-2019 (www.adacore.com/gtkada) with GTK+ 3.24.8 (www.gtk.org) complete,
- Glade 3.22.1 (glade.gnome.org),
- GnatColl 19.2 github.com/AdaCore/gnatcoll),
- Florist mid-2019a (github.com/Blady-Com/florist),
- AdaCurses 6.1 (invisible-island.net/ncurses/ncurses-Ada95.html),
- Gate3 0.5c (sourceforge.net/projects/lorenz),
- Components 4.41 (www.dmitry-kazakov.de/ada/components.htm),
- AICWL 3.21 (www.dmitry-kazakov.de/ada/aicwl.htm),
- Zanyblue 1.4.0 (zanyblue.sourceforge.net),
- PragmARC mid-2019 (pragmada.x10hosting.com/pragmarc.htm),
- GNOGA 1.5-beta (www.gnoga.com),
- AdaControl 1.21r3 (adalog.fr/fr/adacontrol.html),
- AdaDep 1.4r1 (adalog.fr/fr/composants.html),
- AdaSubst 1.6r5 (adalog.fr/fr/composants.html),
- SparForte 2.3-190822 (sparforte.com),

and as side libraries:

- Template Parser 20.0,
- gtksourceview 3.24.4,
- GNUTLS 3.5.18,
- ASIS GPL 2019,
- SDL 1.2.15 et SDL_Image 1.2.12,
- GMP 6.1.2,
- make 4.2.1, NEW
- aspell 0.60.7, NEW
- wget 1.20.3, NEW
- Python 2.7.15,
- Python 3.6.8.

XNAdaLib binaries have been post on Source Forge:

https://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2019-high-sierra/

Feel free to send comments.

Report preferably all comments to MacAda.org mailing list:

http://macada.org/macada/Contacts.html

See list archive:

https://hermes.gwu.edu/archives/gnat-osx.html

Enjoy, Pascal.

http://blady.pagesperso-orange.fr

AdaControl 1.21r6b

From: "J-P. Rosen" <rosen@adalog.fr>
Subject: [Ann] AdaControl 1.21r6 released
Date: Sun, 27 Oct 2019 11:57:05 +0100
Newsgroups: comp.lang.ada

Adalog is pleased to announce the release of version 1.21r6 of AdaControl. This is a bug fix release, with a small addition to the rule Unnecessary_Use_Clause (see HISTORY).

Enjoy!

From: "J-P. Rosen" <rosen@adalog.fr>
Subject: [Ann] Adacontrol v1.21r6b
Date: Wed, 30 Oct 2019 11:13:50 +0100
Newsgroups: comp.lang.ada

There was a minor glitch in the packaging of the version announced two days ago. Please download the correct version, now tagged 1.21r6b.

Sorry for the inconvenience.

http://www.adacontrol.fr

Free-Ada Updated to GCC-9.x

From: Lucretia <laguest9000@googlemail.com>
Subject: Free-Ada updated to GCC-9.x
Date: Wed, 6 Nov 2019 07:32:36 -0800
Newsgroups: comp.lang.ada

[From the project’s website: “This is a set of build scripts to enable you to build the FSF Ada compiler with AdaCore's GPL'd tools.” --arm]

Just an update to let you all know i've added gcc-9.2.0 support to free-ada. I've also added new packages.

https://github.com/Lucretia/free-ada

FYI, branches gcc-9.x and master are equivalent at this stage.

From: Jere <jhb.chat@gmail.com>
Date: Fri, 8 Nov 2019 10:46:23 -0800

Nice! This looks like it only builds in Linux for now? I saw some references to msys in there, but they don't appear to always set the same settings as other host systems. I'm typically doing Ada development from msys2 on Win10 64bit, so was curious about it.

From: Lucretia <laguest9000@googlemail.com>
Date: Fri, 8 Nov 2019 14:38:33 -0800

Yeah, only developed and tested on Linux for now. The aim is to get it working for other platforms, but getting this far on one platform is hard enough.

[...]

Ada Tools for VSCode

From: Lucretia <laguest9000@googlemail.com>
Subject: Ann: VSCode extension - Ada Utilities
Date: Thu, 14 Nov 2019 10:10:17 -0800
Newsgroups: comp.lang.ada

Over the last few days, I've knocked together a VSCode extension to make my life a bit easier, I think others might like it, just copy it in the extensions directory, I've not got it on the marketplace yet.

https://github.com/Lucretia/ada-utilities

From: Jere <jhb.chat@gmail.com>
Date: Thu, 14 Nov 2019 16:01:41 -0800

[Randy Brukardt asked in another post about what VSCode is. --arm]

VSCode is an open source text editor/minimalist IDE (think notepad++ maybe?) provided by microsoft. It has an extension API so developers can add various types of language support and utilities to it (this allows you to customize it to be more like a traditional IDE if you like or do whatever you want with it). I use it a lot for Ada development.

From: briot.emmanuel@gmail.com
Date: Sun, 17 Nov 2019 04:09:39 -0800

[...] the reason [VSCode] is interesting is that it has come with a new protocol named the Language Server Protocol that is used by a lot of editors/IDEs nowadays to do things like cross-reference queries ("go to declaration of", "find all references") to refactoring ("rename an entity"), and more.

To the point that most languages nowadays come with such a server (Ada has one based on libadalang, too). Those servers can be queried from vim (which I use), from Emacs (Stephen has started looking into that for ada-mode, as per a discussion two weeks ago), from GPS (where the Ada language server comes from), Visual Studio Code, and a lot of others.

TclAdaShell Release 20191120

From: Simon Wright <simon@pushface.org>
Subject: ANN: TclAdaShell release 20191120
Date: Wed, 20 Nov 2019 13:38:08 +0000
Newsgroups: comp.lang.ada

This maintenance release is available at Sourceforge.

https://sourceforge.net/projects/tcladashell/files/source/20191120/

Ada and Operating Systems

On Linux Distributions for Ada Programmers

From: reinert <reinkor@gmail.com>
Subject: What is best linux distribution for Ada programmers?
Date: Mon, 12 Aug 2019 23:48:00 -0700
Newsgroups: comp.lang.ada

Debian? Ubuntu?

From: Optikos <optikos@verizon.net>
Date: Tue, 13 Aug 2019 06:25:13 -0700

There was a time …

https://www.youtube.com/watch?v=-3HvUH4fJPM

[Video: Ada in Debian and other distributions, talk by Ludovic Brenta, 2011. --arm]

… when Debian (or all? Debian-derived) was clearly & definitively the correct answer to this question, but Ludovic Brenta doesn't post here at c.l.a anymore, so perhaps his dedication to carrying the Ada torch among the Debian guiding lights might have waned a bit over the years.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 30 Aug 2019 21:18:41 +0200

To be perfectly honest, I have indeed lost most of my energy but my former padawan, Nicolas Boulenguez, has now been a full Debian Developer for several years and continues to update packages (even now doing the transition to gnat-9 for the next Debian release, in experimental). I am very proud of him.

I only occasionally skim this newsgroup and post even less often. But since you're mentioning me I have to express my gratitude and reassure Debian users that Ada is still going strong

IIRC there was a one-man effort to port Ada and many packages to DragonflyBSD, with the express purpose of stealing the crown jewel from Debian. So perhaps DragonflyBSD is a strong contender too nowadays.

A few months ago I held a two-afternoon workshop about cryptography for my 13-year-old son and a few of his friends. Of course we programmed in Ada on Debian. Piece of cake for everyone involved to get GPS and GtkAda running.

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Tue, 15 Oct 2019 19:09:31 -0700

> Debian? Ubuntu?

[...]

But, why would there be a difference between Ubuntu and Debian? The former is very similar to the latter.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 16 Oct 2019 20:43:17 +0200

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

> Ubuntu does their own work keeping Ada up to date?

Not that I know. All they do is import and recompile packages from Debian.

> AFAIK, Ubuntu has GCC 9, Debian is still 8.

Ubuntu is always, by definition and by construction, behind Debian. The compiler is only the foundation of the entire ecosystem of packages. And you should not look at the gcc package but at the gnat package.

From: dirk.dickmanns@gmail.com
Date: Sun, 20 Oct 2019 00:25:18 -0700

> Debian? Ubuntu?

Just for completeness, ArchLinux support is nice with gcc-ada and many additional packages in AUR. I really very much enjoy them, although when regularly updating, there are (the usual) itches and version conflicts (currently clang 8 to 9 for GPS).

From: Lucretia <laguest9000@googlemail.com>
Date: Sun, 17 Nov 2019 04:43:49 -0800

On Sunday, 17 November 2019 10:41:46 UTC, Alain De Vos wrote:

> Two interesting linux distributions with some Ada packages are :

> - fedora

> - gentoo

Gentoo is crap for Ada and is the reason why I started free-ada, so I could have Ada on Gentoo.

Ada and Other Languages

Implementing Rust's Borrow Checked Pointers

From: Lucretia <laguest9000@googlemail.com>
Subject: Implementing Rust's borrow checked pointers
Date: Tue, 24 Sep 2019 02:05:44 -0700
Newsgroups: comp.lang.ada

[I've] been talking to someone on Telegraph and he was saying Ada should implement this, just wondering whether Ada could? I posed a slight change to access type specification to do this, what do people think?

```
type P is restricted access X;
```

Restricted in this case would mean that once assigned it cannot be re-assigned into or out of without some sort of move operation, which could be implemented as an attribute on the access type.

```
A : P := L'Access;
B : P := A'Move; --  A cannot no longer be
                 -- used.
begin
  A.all ... ; -- raises exception.
```

I don't know enough about all this to put a complete proposal together, but I think I've got the basics understood.

From: Optikos <optikos@verizon.net>
Date: Tue, 24 Sep 2019 04:23:28 -0700

[...]

> A.all ... ; -- raises exception.

No, to be as useful as Rust's borrow checker, instead of raising exception, it needs to be a compile-time error. The compiler needs to maintain a whole-program directed graph at compile-time, not defer a detection-based localized analysis to run-time.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Wed, 25 Sep 2019 10:21:46 -0700

> Seems there is an AI for this already.

http://www.ada-auth.org/ai-files/grab_bag/AI12-0240-1v7-stt.TXT

This technique is also now in SPARK, which allows SPARK programs to use access types (in a limited way). See the SPARK reference manual https://docs.adacore.com/

spark2014-docs/html/lrm/
declarations-and-types.html#access-types

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Tue, 24 Sep 2019 18:24:21 +0200

> type P is restricted access X;

> Restricted in this case would mean that once assigned it cannot be re-assigned into or out of without some sort of move operation, which could be implemented as an attribute on the access type.

What's wrong with "limited"?

There's already work on including something like Rust's borrow checking into Ada, but given how rarely access types are needed, and how easy it usually is to confine them to a restricted scope when they are, I don't see that it's worth the effort.

From: Florian Weimer <fw@deneb.enyo.de>
Date: Wed, 25 Sep 2019 18:26:50 +0200

> There's already work on including something like Rust's borrow checking into Ada, but given how rarely access types are needed, and how easy it usually is to confine them to a restricted scope when they are, I don't see that it's worth the effort.

On the other hand, access types are not the only source of unsoundness in Ada. There is at least one other problem, involving records with discriminants with defaults and aliasing.

Pointer Ownership, Containers and Cursors in Ada, Rust, SPARK (cont.)

[See AUJ 40-2 for the initial thread, which discusses how Rust manages to use the borrow checker in complicated structures like containers, and what are its purported advantages. In this subthread, multithreading safety is discussed --arm]

From: "Alejandro R. Mosteo" <alejandro@mosteo.com>
Subject: Re: Microsoft is considering moving to Rust; potential opportunity
Date: Wed, 7 Aug 2019 11:09:05 +0200
Newsgroups: comp.lang.ada

On 6/8/19 19:49, Brad Moore wrote:

> I believe it is also not just related to concurrency. For example, if you pass a pointer to an object into a function which deletes the object, the compiler will detect that use of that pointer after calling the function is not allowed.

Yes; concurrency safety (in a limited sense) is the side-effect, not the main point, I'd say.

From: Jere <jhb.chat@gmail.com>
Date: Wed, 7 Aug 2019 19:13:21 -0700

On Wednesday, August 7, 2019 at 5:09:06 AM UTC-4, Alejandro R. Mosteo wrote:

> Yes; concurrency safety (in a limited sense) is the side-effect, not the main point, I'd say.

I think at some point before the language was first stabilized (2015), both were actively pursued. Here are some musings from the original main developer back in 2013ish [1]. Later on, another Mozilla developer also talked about a similar topic [2].

[1]: http://smallcultfollowing.com/babysteps/blog/2013/06/11/on-the-connection-between-memory-management-and-data-race-freedom/

[2]: https://manishearth.github.io/blog/2015/05/17/the-problem-with-shared-mutability/

It is interesting that both of them kind of hint at the idea of a large single threaded program having similar challenges to a multithreaded program, at least when considering how undefined behavior, data invalidation, and data races occur.

[And here is another reply about self-referencing structures. --arm]

From: "Alejandro R. Mosteo" <alejandro@mosteo.com>
Date: Wed, 7 Aug 2019 11:07:24 +0200

> Although, how does Rust's borrow checker assure the lack of cycles (or assure that the cyclic references are self-contained in a glob that itself has an acyclic reference count, so that the entire glob is condemned en masse)?

It seems you are on your own (e.g. use weak references) to deal with these:

https://doc.rust-lang.org/book/ch15-06-reference-cycles.html

Ada Practice

Single-Character Bugs

[As part of a larger discussion on C++/Fortran bugs, the following subthread emerged. --arm]

From: "J-P. Rosen" <rosen@adalog.fr>
Subject: Re: Mariner 1 / FORTRAN bug
Date: Fri, 9 Aug 2019 08:28:01 +0200
Newsgroups: comp.lang.ada

Le 09/08/2019 à 03:57, robin.vowels@gmail.com a écrit:

>> While it probably didn't cause the failure of a space probe, people did get bitten by this language design flaw, which is an example of a single-character error (added, omitted, or changed) that results in valid code.

> Single-character errors are still possible, whatever the language.

Of course, you can type "-" instead of "+" in any language.

But outside of mathematical formulas, can you give an example of single-character error in Ada?

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Fri, 9 Aug 2019 09:47:11 +0300

I once declared a record type that contained a component, aptly (I thought) called Address, of type System.Address.

Then, in a certain statement dealing with a record object R of that type, I mistakenly wrote

```
R'Address
```

when I meant

```
R.Address
```

Silly me.

I now have a personal coding rule: never use the name Address for a record component of type System.Address. Perhaps this should be expanded to forbid using any record-attribute name as a component name; for example, Size.

From: Maciej Sobczak <see.my.homepage@gmail.com>
Date: Fri, 9 Aug 2019 01:38:12 -0700

> But outside of mathematical formulas, can you give an example of single-character error in Ada?

```
with Ada.Text_IO;
procedure Test is
  procedure P (I, J : Integer) is
  begin
    Ada.Text_IO.Put_Line ("Killing people");
  end;
  procedure P (f : Float) is
  begin
    Ada.Text_IO.Put_Line ("Not killing
              people");
  end;
begin
  P (1,2); -- or should it be P (1.2); ?
end;
```

And of course we have to implement the Initialise operation for our Controlled types, right?

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Fri, 9 Aug 2019 09:27:22 -0500

That is why using named arguments is better and also more clear

```
P (I=>1, J=>2);
```

No chance to mix it up with

```
P (f=>1.2);
```

From: Maciej Sobczak <see.my.homepage@gmail.com>
Date: Fri, 9 Aug 2019 14:05:38 -0700

> That is why using named arguments is better and also more clear

> P (I=>1, J=>2);

Of course - the best way to avoid writing bad code is to write good code. But this is true in any language. What we should expect from good languages is that bad code should be impossible, or at least writing bad code should take more effort than writing good code. And yet, what the above example shows, bad code is perfectly possible in Ada and in fact is easier - and that good code involves higher effort.

Which, ultimately, makes it more difficult for Ada to gain attention of C++ programmers, for example.

To Use or Not to Use Annex E

[After a question on the status of PolyORB, the following comment about the Annex E was made. --arm]

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Subject: Re: polyorb
Date: Tue, 13 Aug 2019 12:09:21 +0200
Newsgroups: comp.lang.ada

On 2019-08-13 10:13, tonyg wrote:

> The flexibility from the Distributed Systems Annex comes from the rigidity of the types used across the partitions. This being a natural amplification from the Ada language.

Annex E is intended and good for tightly coupled static systems, e.g. a network of engine control units of a car.

In a loosely coupled system with nodes going on and off, changing or only modifying their roles and services, RPCs and types known prior to start is not a good choice. Another problem with RPC is that synchronous calls are utterly inefficient and slow. For a real-time system with a time-triggered transport calculated for the worst-case scenario this is no problem. But for most practical applications the load is unpredictable and millisecond accumulating latencies is not an option.

Regarding types, there were many attempts to bring some sort of abstract types and even OO to distributed systems, they all failed (CORBA, ASN.1 included).

This is why data distribution layers stick to some fixed set of primitive types leaving to the application to build upon them. Many have no types at all, only messages (e.g. MQTT). It is not nice but it works.

IMO, annex E's remote types was a good start. But there is a lot of work required to make it full OO, to defining QoS things making it usable in loosely coupled applications.

I also think that current work on new concurrent programming primitives is wasting time. It must be invested into annex E which should serve both concurrent and distributed programming. The difference between a distributed and a multiple core system is not that dramatic (and shared memory architectures will likely die in some future anyway).

Custom Ranges and Predicates

From: Andrew Shvets <andrew.shvets@gmail.com>
Subject: How to best make a custom range?
Date: Mon, 4 Nov 2019 09:26:16 -0800
Newsgroups: comp.lang.ada

Let’s say I have the following code:

```
subtype Test_Char is Character
range ‘A’ .. ‘Z’;
```

But what if I wanted to include ‘&’, ‘@‘ and ‘?’ in this custom range of characters as well? I thought of doing the following, but this obviously failed:

```
subtype Test_Char is Character
range ‘@‘ | ‘&’ | ‘?’ | ‘A’ .. ‘Z’;
```

Or is this impossible unless I use a different approach?

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 4 Nov 2019 11:16:29 -0800

[...]

Try:

```
subtype Upper  is Character range 'A'..'Z';
subtype Lower  is Character range 'a'..'z';
subtype Symbol is Character
with Static_Predicate =>
      Symbol in '&' | '@' | '?';

subtype Test_Character is Character
  with Static_Predicate  => Test_Character
     in Upper | Lower | Symbol,
      Predicate_Failure => Raise
      Constraint_Error;
```

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Tue, 5 Nov 2019 06:02:35 -0800

[...] this is the warning that I get (one of many, but I can’t copy and paste):

```
warning: in instantiation at a-nudira.adb:54
type “Result_Subtype” has predicates,
attribute “First” not allowed
```

From: Shark8 <onewingedshark@gmail.com>
Date: Tue, 5 Nov 2019 07:10:43 -0800

OK, the problem here is that the Ada language *does* discriminate between subtypes with predicates and those without -- mostly because there are arguments about how such attributes should behave. Things like if we have:

```
TYPE Digits is range 0..9;
SUBTYPE Odds is Digits with
   Static_Predicate => Odds in 1|3|5|7|9;
```

what should Odds'Pred(1) be? 0? Constraint_Error?

what about Odds'Pos(1) should it be 0, the first item of the subtype? Or 1, the position in the parent-type?

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Fri, 8 Nov 2019 08:07:22 -0800

Am Freitag, 8. November 2019 16:55:16 UTC+1 schrieb AdaMagica:

> Attributes work on the base type [...]

See RM 3.5(25-27)

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 8 Nov 2019 16:28:56 -0600

> Attributes work on the base type, e.g.

> Natural'Pred (0) = -1

Right, but it's a case-by-case thing as to whether an attribute applies to the type or to the subtype. The ones you're talking about apply to the type (not any subtype), but 'First and 'Last apply to the subtype.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Tue, 5 Nov 2019 18:14:53 +0100

> warning: in instantiation at a-nudira.adb:54

> type “Result_Subtype” has predicates, attribute “First” not allowed

Because ordering attributes are ones that get broken either way. Outside generics the language treats them contravariant [the result is of the base subtype], which breaks ordering but keeps much of other semantics. When you pass a subtype as an actual parameter to a generic it suddenly becomes covariant [the result of the subtype], which is sometimes worse, sometimes quite impossible to implement. Ada plays safe here and just does not let you. In other cases you might not be so lucky. The language cannot deduce right semantics from the constraint. It is undecidable, incomputable etc. In short, do not do that.

Conference Calendar

Dirk Craeynest

KU Leuven, Belgium. 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.

2020

January 14-17 — 12th **Software Quality Days** (SWQD'2020), Vienna, Austria. Topics include: improvement of software development methods, processes, and artifacts; testing and quality assurance of software and software-intensive systems; domain-specific quality issues such as embedded, medical, automotive systems; novel trends in software quality; etc.

January 19-24 — **Summer School on Modelling and Programming Languages** (ModPro'2020), Stellenbosch, South Africa. Includes lecture: "Ensuring System to Software Integrity - From SysML to Simulink to SPARK" by Tucker Taft, AdaCore, USA.

☺ January 19-25 — 47th ACM SIGPLAN **Symposium on Principles of Programming Languages** (POPL'2020), New Orleans, Louisiana, USA.

- January 25 — 1st ACM SIGPLAN **Workshop on Gradual Typing** (WGT'2020). Topics include: integration of compile-time and run-time checking of program invariants, such as integration of static and dynamic type checking.

January 20-22 — 15th **International Conference on High Performance and Embedded Architecture and Compilation** (HiPEAC'2020), Bologna, Italy. Topics include: parallel, multi-core and heterogeneous systems; architectural support for programming productivity; 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; program characterization and analysis techniques; etc.

- ☺ January 21 — **Workshop on Next Generation Real-Time Embedded Systems** (NG-RES'2020). Topics include: programming models, paradigms and frameworks for real-time computation on parallel and heterogeneous architectures; scheduling and schedulability analysis, application of formal methods, compiler-assisted solutions, and middlewares for distributed and/or parallel real-time systems; etc.
- January 21 — 11th **Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures** & 9th **Workshop on Design Tools and Architectures for Multi-Core Embedded Computing Platforms** (PARMA'2020 & DITAM'2020). Topics include: programming models and languages, compilers and virtualization techniques; runtime adaptivity, runtime management, power management and memory management; parallel applications for many-core platforms; etc.

January 20-24 — 46th **International Conference on Current Trends in Theory and Practice of Computer Science** (SOFSEM'2020), Limassol, Cyprus. Topics include: novel and innovative methods and technologies in the broader field of software engineering, including both software product and development process aspects; theories, methods and tools aiming at significantly increasing both the quality of software-intensive systems and the productivity of software development.

☺ January 29-31 — 10th **European Congress on Embedded Real Time Systems** (ERTS'2020), Toulouse, France. Topics include: all aspects of critical embedded real-time systems, such as embedded computing platforms and networked systems, model-based system engineering, formal methods, new programming and verification languages, dependability, safety, cyber security, quality of service, fault tolerance, maintainability, certification, etc.

♦ February 01 — 10th **Ada Developer Room at FOSDEM 2020**. Brussels, Belgium. FOSDEM 2020 is a two-day event (Sat 1 - Sun 2 Feb). This years' edition includes once more a full-day Ada Developer Room, organized by Ada-Belgium in cooperation with Ada-Europe, which will be held on Saturday 1 February.

February 03-07 — 18th **Australasian Symposium on Parallel and Distributed Computing** (AusPDC'2020), Melbourne, Australia. Topics include: all areas of parallel and distributed computing; multi-core systems; GPUs and other forms of special purpose processors; middleware and tools; parallel programming models, languages and compilers; runtime systems; resource scheduling and load balancing; reliability, security, privacy and dependability; etc.

☺ February 18-21 — 40th IEEE **Real-Time Systems Symposium** (RTSS'2019), York, UK. RTSS'2019 was moved from December 3-6 in Hong Kong, to February 18-21 in York, UK. Topics include: all aspects of real-time systems, including theory, design, analysis, implementation, evaluation, and experience.

February 22-23 — 29th ACM SIGPLAN **International Conference on Compiler Construction** (CC'2020), San Diego, California, USA. Co-located with CGO'2020, HPCA'2020, and PPoPP'2020. Topics include: processing programs in the most general sense (analyzing, transforming or executing input programs that describe how a system operates, including traditional compiler construction as a special case); compilation and interpretation techniques (including program representation, analysis, and transformation; code generation, optimization, and synthesis; the verification thereof); run-time techniques (including memory management, virtual machines, ...); programming tools (including refactoring editors, checkers, verifiers, compilers, debuggers, and profilers); techniques for specific domains (such as secure, parallel, distributed, embedded or mobile environments); design and implementation of novel language constructs programming models, and domain-specific languages.

February 27-29 — 13th **Innovations in Software Engineering Conference** (ISEC'2020), Jabalpur, India.

March 11-13 — 28th **Euromicro International Conference on Parallel, Distributed and Network-Based Processing** (PDP'2020), Västerås, Sweden. Topics include: embedded parallel and distributed systems, multi- and many-core systems, GPU and FPGA based parallel systems, programming languages and environments, runtime support systems, performance prediction and analysis, simulation of parallel and distributed systems, shared-memory and message-passing systems, middleware and distributed operating systems, dependability and survivability, real-time distributed applications, etc.

March 11-14 — 51st ACM **Technical Symposium on Computer Science Education** (SIGCSE'2020), Portland, Oregon, USA.

March 16-20 — 25th **International Conference on Architectural Support for Programming Languages and Operating Systems** (ASPLOS-X), Lausanne, Switzerland. Topics include: interplay between programming languages, computer architecture, operating systems, and user interfaces; multicore architectures and systems; programming models, languages, and compilation for all platforms; security, reliability, availability, and sustainability; verification and testing, and their impact on design and security; etc.

☺ March 23-26 — **International Conference on the Art, Science, and Engineering of Programming** (Programming'2020), Porto, Portugal.

> March 23 — 4th **International Workshop on Programming Technology for the Future Web** (ProWeb'2020). Topics include: quality on the new web (static and dynamic program analyses, development tools, automated testing, contract systems, type systems, migration from legacy architectures, API conformance checking, ...); designing for and hosting novel languages on the web; security on the new web; surveys and case studies using state-of-the-art web technology; ideas on and experience reports about how to reconcile the need for quality with the need for agility on the web, how to master and combine the myriad of tier-specific technologies required to develop a web application; etc. Deadline for submissions: January 15, 2020.
>
> March 24 — 4th **Workshop on Modern Language Runtimes, Ecosystems, and VMs** (MoreVMS'2020). Topics include: interoperability between languages, tooling support (e.g. debugging, profiling, etc.), programming language development environments, case studies of existing language implementation approaches, language implementation challenges and trade-offs, surveys and usage reports to understand usage in the wild,

ideas for how we should build languages in the future, etc. Deadline for submissions: January 10, 2020 (extended abstracts, talks).

March 23-27 13th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2020), Porto, Portugal. Topics include: manual testing practices and techniques, security testing, model based testing, test automation, static analysis and symbolic execution, formal verification and model checking, software reliability, testability and design, testing and development processes, testing in specific domains (such as embedded, concurrent, distributed, ..., and real-time systems), testing/debugging tools, empirical studies, experience reports, etc. Deadline for submissions: January 13, 2020 (doctoral symposium).

March 24-27 26th **International Working Conference on Requirements Engineering: Foundation for Software Quality** (REFSQ'2020), Pisa, Italy. Deadline for submissions: January 17, 2020 (workshop papers, doctoral symposium, posters, tools).

Mar 30 - Apr 03 35th ACM **Symposium on Applied Computing** (SAC'2020), Brno, Czech Republic.

☺Mar 30-Apr 03 **Track on Programming Languages** (PL'2020). Topics include: technical ideas and experiences relating to implementation and application of programming languages, such as compiling techniques, domain-specific languages, garbage collection, language design and implementation, languages for modeling, model-driven development, new programming language ideas and concepts, practical experiences with programming languages, program analysis and verification, etc.

Mar 30-Apr 03 **Track on Software Verification and Testing** (SVT'2020). Topics include: new results in formal verification and testing, technologies to improve the usability of formal methods in software engineering, applications of mechanical verification to large scale software, model checking, correct by construction development, model-based testing, software testing, static and dynamic analysis, analysis methods for dependable systems, software certification and proof carrying code, fault diagnosis and debugging, verification and validation of large scale software systems, real world applications and case studies applying software testing and verification, etc.

Mar 30-Apr 03 **Embedded Systems Track** (EMBS'2020). Topics include: system-level design and simulation techniques for embedded systems; testing, debugging, profiling and performance analysis of embedded systems; multicore and SoC-based embedded systems and applications; multithreading in embedded systems design and development; security and dependability support within embedded systems; RTOS for embedded systems, safety-critical embedded systems; compilation strategies, code transformation and parallelization for embedded systems; memory and storage management for embedded systems; case studies; etc.

Mar 30-Apr 03 15th **Track on Dependable, Adaptive, and Secure Distributed Systems** (DADS'2020). Topics include: Dependable, Adaptive, and secure Distributed Systems (DADS); modeling, design, and engineering of DADS; foundations and formal methods for DADS; etc.

♦ April 01-03 20th **International Real-Time Ada Workshop** (IRTAW'2020), Benicàssim, Spain. In cooperation with Ada-Europe. Topics include: review of the Ada 2012 Issues vis-a-vis real-time systems; experiences of use of Ada 2012, in full or by profile, for real-time applications, on single- and multi-processor as well as distributed systems; implementation approaches for Ada 2012 real-time features; review of the Ada 202X Issues with respect to the parallel execution model; early prototyping and use experiences of Ada 202X parallel features, including integration with OpenMP; uses of Ada, possibly with SPARK, in certification-aware domains; review of the Ada language vulnerabilities in relation to TR 24772-2 and -6 by ISO/IEC JTC 1/SC 22/WG 23; contributions to the Ada's conformance test suite for the Real-Time Annex and the upcoming parallel execution model. Deadline for submissions: February 7, 2020 (position papers).

April 06 7th **Workshop on Computational Antifragility and the Engineering of Antifragile Systems** (ANTifragile'2020), Warsaw, Poland. Topics include: dependability, resilience, and antifragile requirements and open issues; design principles, models, and techniques for realizing antifragile systems and behaviours; formal methods for resilience and antifragility; programming language support for resilience and antifragility; specification and verification of resilient and antifragile systems; etc. Deadline for submissions: January 10, 2020.

April 14-17 24th **International Conference on Evaluation and Assessment in Software Engineering** (EASE'2020), Trondheim, Norway. Topics include: assessing the benefits / costs associated with using chosen development technologies; empirical studies using qualitative, quantitative, and mixed methods; evaluation and comparison of techniques and models; replication of empirical studies and families of studies; etc. Deadline for submissions: January 13, 2020 (workshop papers), January 17, 2020 (tutorials, discussion panels).

April 21-24 13th **Cyber-Physical Systems and Internet of Things Week** (CPS Week'2020), Sydney, Australia.

☺ April 21-24 26th IEEE **Real-Time and Embedded Technology and Applications Symposium** (RTAS'2020). Topics include: research related to embedded systems or timing issues, ranging from traditional hard real-time systems to embedded systems without explicit timing requirements, including latency-sensitive systems with informal or soft real-time requirements; original systems and applications, case studies, methodologies, and applied algorithms that contribute to the state of practice in the design, implementation, verification, and validation of embedded systems and time-sensitive systems (of any size); etc.

April 25-30 23rd **European Joint Conferences on Theory and Practice of Software** (ETAPS'2020), Dublin, Ireland. Events include: ESOP (European Symposium on Programming), FASE (Fundamental Approaches to Software Engineering), FoSSaCS (Foundations of Software Science and Computation Structures), TACAS (Tools and Algorithms for the Construction and Analysis of Systems). Deadline for submissions: January 19, 2020 (nominations ETAPS doctoral dissertation award).

☺ April 25-30 **VerifyThis Verification Competition 2020**. Topics include: VerifyThis Collaborative Long-Term Challenge; no restrictions on verification technology used; at least one team is using SPARK.

☺ April 26 12th **Workshop on Programming Language Approaches to Concurrency- and communication-cEntric Software** (PLACES'2020). Topics include: general area of programming language approaches to concurrency, communication, and distribution and may range from foundational issues, language implementations, to applications and case studies; design and implementation of programming languages with first class concurrency and communication; models, such as process algebra and automata; concurrent data types, objects, and actors; verification and program analysis methods for concurrent and distributed software; etc. Deadline for submissions: January 24, 2020 (abstracts), February 28, 2020 (papers).

April 27-30 15th **European Conference on Computer Systems** (EuroSys'2020), Heraklion, Crete, Greece. Topics include: all areas of computer systems research, such as distributed systems, language support and runtime systems, systems security and privacy, dependable systems, parallelism, concurrency, and multicore systems, real-time, embedded, and cyber-physical systems, tracing, analysis, verification, and transformation of systems, etc. Deadline for submissions: February 21, 2020 (doctoral workshop).

May 04-08 23rd **Ibero-American Conference on Software Engineering** (CIbSE'2020), Curitiba, Brazil. Deadline for submissions: February 4, 2020 (doctoral symposium, industry talks).

May 11-13 ACM **International Conference on Computing Frontiers** 2020 (CF'2020), Catania, Sicily, Italy. Topics include: embedded, IoT and cyber-physical systems; large-scale system design and networking; system software, compiler technologies and programming languages; fault tolerance and resilience; security; etc. Deadline for submissions: January 28, 2020.

May 11-15 12th **NASA Formal Methods Symposium** (NFM'2020), Moffett Field, California, USA. Topics include: identifying challenges and providing solutions towards achieving assurance for critical systems; formal verification, including theorem proving, model checking, and static analysis; run-time verification; techniques and algorithms for scaling formal methods, such as abstraction and symbolic methods, compositional techniques, as well as parallel and/or distributed techniques; safety cases and system safety; formal approaches to fault tolerance; design for verification and correct-by-design techniques; empirical evaluations of formal methods techniques for safety-critical systems; formal methods in systems engineering and model-based development; etc.

☺ May 19-21 23rd IEEE **International Symposium On Real-Time Distributed Computing** (ISORC'2020), Nashville, Tennessee, USA. Topics include: object/component/service-oriented real-time distributed computing (ORC) technology; software architectures for real-time distributing computing (programming paradigms, ORC paradigms, object/component models, languages, synchronous languages), trusted and dependable systems, system software (real-time kernels, operating systems, distribution middleware for ORC, extensibility, synchronization, resource allocation, scheduling, timing analysis, fault tolerance and resilience, security, ...), applications (medical devices, intelligent transportation systems, industrial automation systems and Industry 4.0, Internet of Things and Smart Grids, embedded and cyber-physical systems in automotive, avionics, autonomous vehicles, consumer electronics, ...), system evaluation (performance & timing evaluation, dependability, fault detection and recovery time, ...), etc. Deadline for submissions: January 17, 2020 (main track), March 24, 2020 (posters, demos).

May 23-29 42nd **International Conference on Software Engineering** (ICSE'2020), Seoul, South Korea. Topics include: the full spectrum of Software Engineering.

May 23-29 **Software Engineering Education and Training** (SEET'2020). Topics include: novel methods of teaching software engineering skills, empirical studies describing software engineering education contexts, novel learning technologies that support software engineering education and training, well-substantiated arguments about what skills are most essential to learn, etc.

May 25-26 3rd **International Conference on Technical Debt** (TechDebt'2020). Topics include: the business case for technical debt management; understanding causes and effects of technical debt; technical debt management within software life-cycle management; technical debt in design and architecture; technical debt and software evolution, maintenance, and aging; concrete practices and tools used to manage technical debt; debt remediation and refactoring; technical debt and quality attributes, such as security (especially at run-time); technical debt in (ultra-) large-scale systems, ecosystems, platforms and product lines; success and failure stories of technical debt management; education and training related to technical debt; etc. Deadline for submissions: January 10, 2020 (abstracts), January 17, 2020 (papers). Deadline for early registration: March 17, 2020.

May 25 8th **International Conference on Formal Methods in Software Engineering** (FormaliSE'2020). Topics include: approaches and tools for verification and validation; application of formal methods to specific domains, e.g. autonomous, cyber-physical, and IoT systems; scalability of formal methods applications; integration of formal methods within the software development lifecycle formal specification; model-based engineering approaches; formal methods in a certification context; formal approaches for safety and security-related issues; usability of formal methods; guidelines to use formal methods in practice; case studies developed/analyzed with formal approaches; experience reports on the application of formal methods to real-world problems; etc. Deadline for submissions: January 9, 2020 (abstracts), January 16, 2020 (papers).

May 25-26 23rd **International Workshop on Software and Compilers for Embedded Systems** (SCOPES'2020), St. Goar, Germany. Topics include: all aspects of compilation and mapping process of embedded systems, such as models of computation and programming languages; automatic code parallelization techniques; mapping and scheduling techniques for embedded multi-processor systems; code generation techniques for embedded single- and multi-processor architectures; design of real-time systems; techniques for compiler aided profiling, measurement, debugging and validation of embedded software; etc. Deadline for submissions: February 14, 2020 (papers).

June 03-05 20th **International Conference on Computational Science** (ICCS'2020), Amsterdam, the Netherlands. Topics include: scientific computing, complex systems - modelling and simulation, parallel and distributed computing, new programming models, education in computational science, etc. Deadline for submissions: January 6, 2020 (papers).

♦ June 08-12 25th **Ada-Europe International Conference on Reliable Software Technologies** (AEiC 2020 aka Ada-Europe 2020), Santander, Spain. Sponsored by Ada-Europe, in cooperation with ACM SIGAda (pending), and the Ada Resource Association (ARA). Deadline for submissions: January 14, 2020 (journal-track papers, industrial presentation outlines, tutorial and workshop proposals), 31 March 2020 (Work-in-Progress papers).

June 08-12 — 21st **International Conference on Agile Software and Systems Development** (XP'2020), Copenhagen, Denmark. Deadline for submissions: January 17, 2020 (experience reports), January 29, 2020 (research paper abstracts), February 5, 2020 (research papers, Industry and Practice, Agile in Education and Training), February 20, 2020 (on-site research, Diversity & Inclusion in Agile, journal first), March 23, 2020 (doctoral symposium).

☺ June 09-11 — 28th **International Conference on Real-Time Networks and Systems** (RTNS'2020), Paris, France. Topics include: real-time applications design and evaluation (automotive, avionics, space, railways, telecommunications, process control, multimedia), real-time aspects of emerging smart systems (cyber-physical systems and emerging applications, ...), real-time system design and analysis (real-time tasks modeling, task/message scheduling, mixed-criticality systems, Worst-Case Execution Time (WCET) analysis, ...), software technologies for real-time systems (model-driven engineering, programming languages, compilers, WCET-aware compilation and parallelization strategies, middleware, Real-time Operating Systems (RTOS), hypervisors, ...), formal specification and verification, real-time distributed systems (fault tolerance, publisher/subscriber protocols, ...), etc.

☺ June 16 — 21st ACM SIGPLAN/SIGBED **International Conference on Languages, Compilers, and Tools for Embedded Systems** (LCTES'2020), London, UK. Co-located with PLDI'2020. Topics include: support for enhanced programmer productivity; support for enhanced debugging, profiling, and exception/interrupt handling; hardware, system software, application software, and their interfaces; system integration and testing; run-time system support for embedded systems; support for system security and system-level reliability; validation and verification, in particular of concurrent and distributed systems; formal foundations of model-based design for code generation, analysis, verification; architecture for new language features, virtualization, compilation, debugging tools; empirical studies and their reproduction, and confirmation; etc. Deadline for submissions: February 28, 2020.

June 18-19 — **European Conference on Software Engineering Education** (ECSEE'2020), Seeon Monastery, Bavaria, Germany. Deadline for submissions: March 16, 2020.

June 22-26 — **Software Technologies: Applications and Foundations** (STAF'2020), Bergen, Norway.

> June 22-26 — 14th **International Conference on Tests And Proofs** (TAP'2020). Topics include: many aspects of verification technology, including foundational work, tool development, and empirical research; the connection between proofs (and other static techniques) and testing (and other dynamic techniques); verification and analysis techniques combining proofs and tests; program proving with the aid of testing techniques; deductive techniques supporting the automated generation of test vectors and oracles; deductive techniques supporting novel definitions of coverage criteria; program analysis techniques combining static and dynamic analysis; testing and runtime analysis of formal specifications; verification of verification tools and environments; applications of test and proof techniques in new domains, such as security, configuration management, learning; combined approaches of test and proof in the context of formal certifications (Common Criteria, CENELEC, ...); case studies, tool and framework descriptions, and experience reports about combining tests and proofs; etc. Deadline for submissions: January 15, 2020 (abstracts), January 22, 2020 (papers).

☺ July 13-17 — 34th **European Conference on Object-Oriented Programming** (ECOOP'2020), Berlin, Germany. Topics include: design, implementation, optimization, analysis, and theory of programs, programming languages, and programming environments. Deadline for submissions: January 10, 2020 (papers).

July 13-17 — 44th **Annual** IEEE **Conference on Computers, Software and Applications** (COMPSAC'2020), Madrid, Spain. Deadline for submissions: January 20, 2020 (main conference papers), April 9, 2020 (worskhop papers).

July 29-31 — 32nd **International Conference on Software Engineering Education and Training** (CSEET'2020), Munich, Germany. Topics include: Teaching formal methods (TFM), Teaching "real world" SE practices (TRW), Software quality assurance education (SQE), Global and distributed SE education (GDE), Open source in education (OSE), Cooperation between Industry and Academia (CIA), Training models in industry (TMI), Continuous education (CED), Methodological aspects of SE education

(MAE), etc. Deadline for submissions: February 1, 2020 (abstracts for research papers, industrial experience reports, Journal First submissions, posters, tools, panels), February 8, 2020 (research papers, industrial experience reports, Journal First submissions, posters, tools, panels).

☺ Sep 02-03 — 25th **International Conference on Formal Methods for Industrial Critical Systems** (FMICS'2020), Vienna, Austria. Co-located with CONCUR'2020 and FORMATS'2020. Topics include: case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; methods, techniques and tools to support automated analysis, certification, debugging, descriptions, learning, optimisation and transformation of complex, distributed, real-time, embedded, mobile and autonomous systems; verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability (e.g., scalability and usability issues); impact of the adoption of formal methods on the development process and associated costs; application of formal methods in standardisation and industrial forums. Deadline for submissions: May 8, 2020 (abstracts), May 15, 2020 (papers).

September 09-11 — 13th **International Conference on the Quality of Information and Communications Technology** (QUATIC'2020), Faro, Portugal. Topics include: all quality aspects in ICT systems engineering and management; quality in ICT process, product and applications domains; practical studies; etc. Tracks on ICT verification and validation, safety, security and privacy, model-driven methods, agile methods, evolution in ICT / reengineering and refactoring, evidence-based software quality engineering, software quality education and training, etc. Deadline for submissions: March 30, 2020 (full papers), May 25, 2020 (short papers).

September 15-18 — 39th **International Conference on Computer Safety, Reliability and Security** (Safecomp'2020), Lisbon, Portugal. Topics include: all aspects related to the development, assessment, operation and maintenance of safety-related and safety-critical computer systems; formal modelling, verification and validation; model-driven engineering; security and privacy protection mechanisms; safety/security co-engineering and risk assessment; testing, verification and validation methods & tools; qualification, assurance and certification methods & tools; cyber-physical threats and vulnerability analysis; safety and security guidelines, standards and certification; etc. Domains of application include: railways, automotive, space, avionics & process industries; highly automated and autonomous systems; telecommunication and networks; safety-related applications of smart systems and IoT; critical infrastructures,; medical devices and healthcare; surveillance, defense, emergency & rescue; logistics, industrial automation, off-shore technology; education & training; etc. Deadline for submissions: February 7, 2020 (workshops), February 14, 2020 (abstracts), February 24, 2020 (full papers).

September 22-24 — 19th **International Conference on Intelligent Software Methodologies, Tools and Techniques** (SOMET'2020), Kytakyushu, Japan. Topics include: state-of-art and new trends on software methodologies, tools and techniques; software methodologies, and tools for robust, reliable, non-fragile software design; software developments techniques and legacy systems; automatic software generation versus reuse, and legacy systems; software evolution techniques; Agile Software and Lean Methods; formal methods for software design; software maintenance; software security tools and techniques; formal techniques for software representation, software testing and validation; software reliability; Model Driven Development (DVD), code centric to model centric software engineering; etc. Deadline for submissions: March 20, 2020 (papers).

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

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

Call for Participation
10th Ada Developer Room at FOSDEM 2020
Saturday 1 February 2020, Brussels, Belgium

Organized by Ada-Belgium
in cooperation with Ada-Europe

FOSDEM[1], the Free and Open source Software Developers' European Meeting, is a non-commercial two-day weekend event organized early each year in Brussels, Belgium. It is highly developer-oriented and brings together 8000+ participants from all over the world. The 2020 edition takes place on Saturday 1 and Sunday 2 February. It is free to attend and no registration is necessary.

In this edition, Ada-Belgium[2] organizes once more a full day with 8.5 hours of presentations related to Ada and Free or Open Software in a s.c. Developer Room. The "Ada DevRoom" at FOSDEM 2020 is held on the first day of the event. The program offers introductory presentations on the Ada programming language, as well as more specialised presentations on focused topics, tools and projects. This year FOSDEM has a total of 13 Ada-related presentations by 12 authors from 8 countries!

Program overview:

- *Welcome to the Ada DevRoom*, by Dirk Craeynest, Ada-Belgium
- *An Introduction to Ada for Beginning and Experienced Programmers*, by Jean-Pierre Rosen, Adalog, France
- *HAC: the Compiler which will Never Become Big*, by Gautier de Montmollin, Ada-Switzerland
- *Tracking Performance of a Big Application from Dev to Ops*, by Philippe Waroquiers, Eurocontrol, Belgium
- *Cappulada: What we've Learned*, by Johannes Kliemann, Componolit, Germany
- *Programming ROS2 Robots with RCLAda*, by Alejandro R. Mosteo, Centro Universit. de la Defensa, Spain
- *Live Demo of Ada's Distribution Features*, by Jean-Pierre Rosen, Adalog, France
- *Writing Shared Memory Parallel Programs in Ada*, by Jan Verschelde, Univ. of Illinois at Chicago, USA
- *Spunky: a Genode Kernel in Ada/SPARK*, by Martin Stein, Genode Labs, Germany
- *Alire: Ada Has a Package Manager*, by Alejandro R. Mosteo, Centro Universitario de la Defensa, Spain, Pierre-Marie de Rodat and Fabien Chouteau, AdaCore, France
- *Protect Sensitive Data with Ada Keystore*, by Stephane Carrez, Twinlife, France
- *EUgen: a European Project Proposal Generator*, by Riccardo Bernardini, University of Udine, Italy
- *On Rapid Application Development in Ada*, by Tomasz Maluszycki, Poland
- *Ada-TOML: a TOML Parser for Ada*, by Pierre-Marie de Rodat, AdaCore, France

The Ada at FOSDEM 2020 web-page has all details, such as the full schedule, abstracts of presentations, biographies of speakers, and pointers to more info. For the latest information at any time, contact <Dirk.Craeynest@cs.kuleuven.be>, or see:

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem.html

1https://fosdem.org/2020
2http://www.cs.kuleuven.be/~dirk/ada-belgium

20th International Real-Time Ada Workshop – IRTAW 2020

Hotel Voramar, Benicàssim, Spain
1-3 April 2020
http://www.ada-europe.org/irtaw/2020

Call for Papers

Since its inception, the International Real-Time Ada Workshop (IRTAW) series has provided a forum where members of the research, user, and implementer communities have raised issues with the real-time systems support in Ada, and explored possible solutions to them, and approaches to evolve those aspects of the language to the new frontiers of real-time computing. Over the years, the IRTAW has made important contributions to the 2005 and 2012 revisions of the Ada programming language standard, for the tasking features, the real-time and high-integrity systems annexes, the Ravenscar and the Jorvik Profiles, and – more recently – the parallel execution model of the language due to appear in the 202X revision.

The topics of interest to the 20th edition of IRTAW include but are not limited to:

- Review of the Ada 2012 Issues *vis-a-vis* real-time systems;
- Experiences of use of Ada 2012, in full or by profile, for real-time applications, on single- and multi-processor as well as distributed systems;
- Implementation approaches for Ada 2012 real-time features;
- Review of the Ada 202X Issues with respect to the parallel execution model;
- Early prototyping and use experiences of Ada 202X parallel features, including integration with OpenMP;
- Uses of Ada, possibly with SPARK, in certification-aware domains;
- Review of the Ada language vulnerabilities in relation to TR 24772-2 and -6 by ISO/IEC JTC 1/SC 22/WG 23;
- Contributions to the Ada's conformance test suite for the Real-Time Annex and the upcoming parallel execution model.

Participation in the 20th IRTAW is by invitation, following the submission of a position paper that addresses pertinent topics. Anyone unable to produce a position paper, but still wishing to receive an invitation, is encouraged to send a one-page position statement, in PDF, to the Program Chair, indicating their interests and reasons for attending.
Priority in selection will be given to the authors of submitted papers.

Submission Requirements

Position papers should not exceed ten pages in typical IEEE conference layout, excluding code snippets. They shall be submitted, in PDF, to the Program Chair at the email address listed below. All accepted papers will appear, in their final form after the event, in a special issue of Ada Letters (ACM Press) also reprinted by the Ada User Journal. Authors with a relevant paper submitted to the Ada-Europe 25th International Conference on Reliable Software Technologies, AEiC 2020 (deadline 7 January, 2020) may offer an extended abstract of the same material to IRTAW.

Important Dates

Paper Submission: **7 February, 2020**
Notification of Acceptance: 6 March, 2020
Confirmation of Attendance: 13 March, 2020
Final Paper Due: 23 March, 2020
Workshop: **1-3 April, 2020**

Program Chair
Tullio Vardanega, *University of Padova, Italy*
`tullio.vardanega@unipd.it`

Workshop Chair
Jorge Real, *Universitat Politècnica de València, Spain*
`jorge@disca.upv.es`

25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2020)

8-12 June 2020, Santander, Spain

(C) RMR

Conference Chair
Michael González Harbour
Universidad de Cantabria, Spain
mgh@unican.es

Program Chair
Mario Aldea Rivas
Universidad de Cantabria, Spain
aldeam@unican.es

Work-in-Progress Chair
Kristoffer Nyborg Gregertsen
SINTEF Digital, Norway
kristoffer.gregertsen@sintef.no

Tutorial & Workshop Chair
Jorge Garrido Balaguer
Universidad Politécnica de Madrid, Spain
jorge.garrido@upm.es

Industrial Chair
Patricia Balbastre Betoret
Universitat Politècnica de València, Spain
patricia@ai2.upv.es

Exhibition & Sponsorship Chair
Ahlan Marriott
White Elephant GmbH, Switzerland
software@white-elephant.ch

Publicity Chair
Dirk Craeynest
Ada-Belgium & KU Leuven, Belgium
dirk.craeynest@cs.kuleuven.be

In cooperation with:

(pending)

General Information

The 25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2020 aka Ada-Europe 2020) will take place in Santander, Spain. The conference schedule includes a technical program, vendor exhibition and parallel tutorials and workshops.

The 2020 edition of the conference continues the major revamp in the registration fees introduced in 2019, redesigned to extend participation from industry and academia, and to reward contributors, especially but not solely, students and post-doc researchers.

Schedule

14 January 2020	Submission of journal-track papers, industrial presentation outlines, and tutorial and workshop proposals (EXTENDED)
20 March 2020	Notification of acceptance for journal-track papers, industrial presentations, tutorials and workshops
31 March 2020	Submission of Work-in-Progress (WiP) papers
30 April 2020	Notification of acceptance for WiP papers

Topics

The conference is 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 from industry, academia and government organizations active in the promotion and development of reliable software technologies.

The topics of interest for the conference include but are not limited to:

- Design and Implementation of Real-Time and Embedded Systems,
- Design and Implementation of Mixed-Criticality Systems,
- Theory and Practice of High-Integrity Systems,
- Software Architectures for Reliable Systems,
- Methods and Techniques for Quality Software Development and Maintenance,
- Ada Language and Technologies,
- Mainstream and Emerging Applications with Reliability Requirements,
- Achieving and Assuring Safety in Machine Learning Systems,
- Experience Reports on Reliable System Development,
- Experiences with Ada.

Refer to the conference website for the full list of topics.

www.ada-europe.org/conference2020

(C) Pachi Hondal

Program Committee

Mario Aldea, Univ. de Cantabria, ES
Iain Bate, University of York. UK
Johann Blieberger, Vienna Univ. of Technology, AT
Bernd Burgstaller, Yonsei Univ., KR
Daniela Cancila, CEA LIST, FR
António Casimiro, Univ. Lisboa, PT
Juan A. de la Puente, Univ. Pol. de Madrid, ES
Barbara Gallina, Mälardalen Univ., SE
Marisol García Valls, Univ. Pol. de València, ES
J. Javier Gutiérrez, Univ. de Cantabria, ES
Jérôme Hugues, CMU/SEI, USA
Hubert Keller, Karlsruhe Institute of Technology, DE
Patricia López Martínez, Univ. de Cantabria, ES
Kristoffer Nyborg Gregertsen, SINTEF Digital, NO
Laurent Pautet, Telecom ParisTech, FR
Luís Miguel Pinho, CISTER/ISEP, PT
Erhard Plödereder, Univ. Stuttgart, DE
Jorge Real, Univ. Pol. de València, ES
José Ruiz, AdaCore, FR
Sergio Sáez, Univ. Pol. de València, ES
Frank Singhoff, Univ. de Bretagne Occidentale, FR
Tucker Taft, AdaCore, USA
Elena Troubitsyna, Åbo Akademi Uni., FI
Santiago Urueña, GMV, ES
Tullio Vardanega, Univ. of Padua, IT
Eugenio Villar Bonet, Univ. de Cantabria, ES

Industrial Committee

Ian Broster, Rapita Systems, UK
Javier Coronel, FentISS, ES
Dirk Craeynest, Ada-Belgium & KU Leuven, BE
Thomas Gruber, Austrian Institute of Technology (AIT), AT
Ismael Lafoz, Airbus Defence and Space, ES
Ahlan Marriott, White Elephant, CH
Maurizio Martignano, Spazio-IT, IT
Laurent Rioux, Thales R&T, FR
Marian Roselló, Terma, NL
Jean-Pierre Rosen, Adalog, FR
Emilio Salazar, GMV, ES

Call for Journal-Track Papers

The **journal-track papers** submitted to the conference are full-length papers that must describe mature research work on the conference topics. They must be original and shall undergo anonymous peer review.

Accepted journal-track papers will get a presentation slot within a technical session of the conference and they will be published in an open-access special issue of the **Journal of Systems Architecture** with no additional costs to authors.

The corresponding authors shall submit their work by 14 January 2020 via the Special Issue web page: *https://www.journals.elsevier.com/journal-of-systems-architecture/call-for-papers/advances-in-reliable-software-technologies*

Submitted papers must follow the guidelines provided in the "Guide-for-Authors" of the JSA (*https://www.elsevier.com/journals/journal-of-systems-architecture/1383-7621/guide-for-authors*). In particular, JSA does not impose any restriction on the format or extension of the submissions.

Call for WiP-Track Papers

The **Work-in-Progress papers (WiP-track)** are short (4-page) papers describing evolving and early-stage ideas or new research directions. They must be original and shall undergo anonymous peer review. The corresponding authors shall submit their work by 31 March 2020, via *https://easychair.org/conferences/?conf=aeic2020*, strictly in PDF and following the Ada User Journal style (*http://www.ada-europe.org/auj/*).

Authors of accepted WiP-track papers will get a presentation slot within a regular technical session of the conference and will also be requested to present a poster. The papers will be published in the Ada User Journal as part of the proceedings of the Conference.

The conference is listed in the principal citation databases, including DBLP, Scopus, Web of Science, and Google Scholar. The Ada User Journal is indexed by Scopus and by EBSCOhost in the Academic Search Ultimate database.

Call for Industrial Presentations

The conference seeks **industrial presentations** that deliver insightful information value but may not sustain the strictness of the review process required for regular papers. The authors of industrial presentations shall submit their proposals, in the form of a short (one or two pages) abstract, by 14 January 2020, via *https://easychair.org/conferences/?conf=aeic2020*, strictly in PDF and following the Ada User Journal style (*http://www.ada-europe.org/auj/*).

The Industrial Committee will review the submissions anonymously and make recommendations for acceptance. The abstract of the accepted contributions will be included in the conference booklet, and authors will get a presentation slot within a regular technical session of the conference.

These authors will also be invited to expand their contributions into articles for publication in the Ada User Journal, as part of the proceedings of the Industrial Program of the Conference.

Awards

Ada-Europe will offer an honorary award for the best presentation.

Call for Educational Tutorials

The conference is seeking tutorials in the form of educational seminars including hands-on or practical demonstrations. Proposed tutorials can be from any part of the reliable software domain, they may be purely academic or from an industrial base making use of tools used in current software development environments. We are also interested in contemporary software topics, such as IoT and artificial intelligence and their application to reliability and safety.

Tutorial proposals shall include a title, an abstract, a description of the topic, an outline of the presentation, the proposed duration (half day or full day), and the intended level of the tutorial (introductory, intermediate, or advanced). All proposals should be submitted by e-mail to the Educational Tutorial Chair.

The authors of accepted full-day tutorials will receive a complimentary conference registration. For half-day tutorials, this benefit is 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 days. Workshop proposals should be submitted by e-mail to the Workshop Chair. The workshop organizer shall also commit to producing the proceedings of the event, for publication in the Ada User Journal.

Call for Exhibitors

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

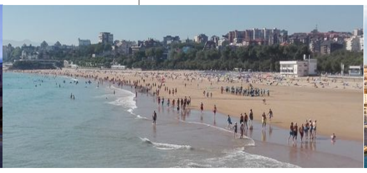
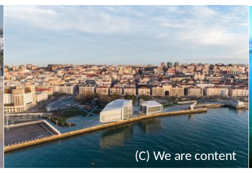
Special Registration Fees

Authors of accepted contributions and all students will enjoy reduced registration fees.

Venue

Santander is a nice tourist city in the north of Spain, with a well-connected airport and at a 100 km drive from Bilbao airport.

The conference venue and hotel is the Bahia Hotel in the city center and beside Santander bay.

VECTOR

Automate Your Ada Unit Testing

With VectorCAST/Ada

VectorCAST/Ada is an integrated software test solution that significantly reduces the time, effort, and cost associated with testing Ada software components necessary for validating safety- and mission-critical embedded systems.

> Complete test-harness construction for unit and integration testing
> Test execution from GUI or scripts
> Code coverage analysis
> Regression Testing
> Code complexity calculation
> Automatic test creation based on decision paths
> User-defined tests for requirements-based testing
> Test execution trace and playback to assist in debugging
> Integrations with best of breed requirements traceability tools

More information: **www.vector.com/vectorcast**

Vector Austria GmbH | www.vector.com

Co-engineering of Safety and Security Life Cycles for Engineering of Automotive Systems

Robert Bramberger and Helmut Martin

Virtual Vehicle Research GmbH, Graz, Austria, {Robert.Bramberger, Helmut.Martin}@v2c2.at

Barbara Gallina

Mälardalen University, Västerås, Sweden; Barbara.Gallina@mdh.se

Christoph Schmittner

AIT Austrian Institute of Technology GmbH, Vienna, Austria; Christoph.Schmittner@ait.ac.at

Abstract

Nowadays systems are becoming more and more connected. Consequently, the co-engineering of (cyber)security and safety life cycles becomes paramount. Currently, no standard provides a structured co-engineering process to facilitate the communication between safety and security engineers. In this paper, we propose a process for co-engineering safety and security by the explicit systematization and management of commonalities and variabilities, implicitly stated in the requirements of the different standards. Our process treats the safety and security life cycles as members of a security-informed safety-oriented process line and so it forces safety and security engineers to come together and brainstorm on what might be considered a commonality and what might be considered a variability. We illustrate the usage of our process by systematizing commonalities and variabilities at risk analysis phase in the context of ISO 26262 and SAE J3061. We then draw lessons learnt. Finally, we sketch some directions for future work.

Keywords: Security-informed Safety, ISO 26262, SAE J3061, Security-informed Safety-oriented Process Line Engineering (SiSoPLE), HARA, TARA

1 Introduction

Nowadays, systems are becoming more and more connected and offer advanced functionalities. In the automotive domain, for instance, with the advent of Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I), and even vehicle to cloud (V2C) communication, road vehicles are playing an active and major role within the Internet of Things (IoT), offering new communication-centred functionalities aimed at increasing safety by e.g., decongesting traffic via roadworks-related communication. However, connectivity may threaten safety, due to numerous security threats, which, as recently surveyed by ENISA [3], are emerging.

Consequently, the co-engineering of (cyber)security and safety life cycles becomes paramount. Currently, no standard provides a structured co-engineering process to facilitate the communication between safety and cybersecurity engineers. ISO 26262 [5] introduces a standardized safety life cycle, which needs to be complemented by requirements stemming from cybersecurity standards (e.g. the upcoming cybersecurity standard ISO/ SAE 21434 [6]) and/or guidelines (e.g. SAE J3061 [8]). SAE J3061 is the only published guidebook that provides suggestions for considering both concerns. Specifically, SAE J3061 proposes a life cycle for handling cybersecurity which is based on the ISO 26262 safety lifecycle. The reason for this analogous life cycle is to allow organizations with safety processes based on ISO 26262 to use a **common framework for cybersecurity and safety** to facilitate the development of a tailored cybersecurity process by capitalizing on aspects of an organization's existing safety process that are common to both cybersecurity and safety, for example, the supporting process procedures and templates.

Thus, the co-engineering of safety and (cyber)security life cycles and, more broadly, the co-engineering of different mono-concern life cycles can be facilitated by the explicit systematization and management of commonalities and variabilities, implicitly stated in the requirements of the different standards. This leads to the engineering of a Security-informed Safety-oriented Process Line (SiSoPL) [13].

In this paper, we extend our initial thoughts published in [16], [29] and we engineer an automotive SiSoPL by using the tool-chain constituted of Eclipse Process Framework Composer (EPF-C) [33]. EPF-C permits users to engineer processes in compliance with a SPEM (Software & Systems Process Engineering Metamodel) 2.0-like language [30], and BVR Tool [31], which permits users to orthogonally manage variability at process level in compliance with the Base Variability Resolution (BVR) language [18]. The toolchain (shown in [21]) obtained via the integration between EPF-C and BVR Tool is part of the AMASS tool platform, delivered by the AMASS project [1], [26], [28], [29] and hosted by OpenCert [32]. The engineered SiSoPL embraces the automotive regulations comprising ISO 26262 and SAE J3061 and focuses on the risk analysis phase, as initially done in [12], where the automotive Security-informed Safety terminological framework for retrieving the implicit commonalities was proposed. Finally, from our engineered SiSoPL, we derive a single security-informed safety-oriented process targeting the security-informed safety concept of a car2car communication management unit.

The rest of the paper is organized as follows. In Section 2, we recall essential background. In Section 3, we clearly state the problem. In Section 4, we explain our methodology for SiSoPL engineering. In Section 5, we apply our methodology and report about our lessons learned. Finally, in Section 6, we draw our concluding remarks and sketch future work.

2 Background and related work

In this section, we present the background information on which we base our work.

2.1 Relevant safety and cybersecurity standards

In this sub-section, we provide an overview about the standards for functional safety and cybersecurity targeting non-autonomous road vehicles.

ISO 26262 [5] is the automotive functional safety standard (first release 2011). It describes a safety life cycle for the development of safety-related automotive systems with the purpose of guaranteeing absence of unreasonable risk due to hazards caused by malfunctioning behaviour of electrical/electronic systems. The scope in edition 2018 has been extended from passenger cars to further road vehicles. Now it also deals with trucks, busses and motorcycles. ISO 26262 provides an informative guideline on “potential interaction of functional safety with cybersecurity”.

The life cycle is structured into phases. The first phase, called the concept phase, starts with the item definition, i.e., a description of the system with regard to its functionality, interfaces, environment, etc. Once the item is defined, the HARA (Hazard Analysis and Risk Assessment) is performed to identify/categorize/evaluate hazardous events, i.e., the combination of hazard (potential source of harm) and operational situations (scenarios that can occur during vehicle’s life). Harm is defined as the physical injury or damage to the health of persons. To minimize harm, unreasonable risk has to be reduced. To support risk evaluation, ISO 26262 has introduced the notion of Automotive Safety Integrity Level (ASIL), which can assume one out of five values, ranging from negligible QM and ASIL A to ASIL D, where D represents the most stringent level. An ASIL is assigned based on the severity, the exposure, and the controllability of the hazardous event. The assignment of the ASIL constrains the stringency of the following activities within the safety life cycle. Another parameter used to influence the stringency is the recommendation level (neutral, recommended, highly recommended), abbreviated as RecL, which is typically assigned in conjunction with the ASIL to provide guidance on method application.

The result of the concept phase is the functional safety concept, represented by the set of safety goals (top-level safety requirements) derived from the HARA findings.

SAE J3061 [8] provides high level guiding principles for cybersecurity for the complete engineering life cycle. It proposes more concrete communication paths between functional safety and cybersecurity engineering.

The cybersecurity life cycle initiates at the concept phase with the feature (i.e., system) definition in which the scope of the feature is specified with respect to physical boundaries, cybersecurity perimeter, and trust boundaries of the feature. After that, TARA (Threat Analysis and Risk Assessment) is performed. TARA is an analysis technique applied to identify potential threats to a feature and to assess the risk associated with the identified threats. Cybersecurity goals are derived and formulated for each of the highest risk potential threats documented in the TARA. SAE J3061 does not introduce a specific notion for cybersecurity level. However, it outlines a sampling of security analysis methods for performing the TARA such as the method used by the E-Safety Vehicle Intrusion Protected Applications (EVITA) program, the Threat, Vulnerabilities, and implementation Risks Analysis (TVRA) method, the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) method, and the HEAling Vulnerabilities to ENhance Software Security and Safety (HEAVENS) method and attack tree information. These methods propose possible security levels.

ISO/SAE 21434 defines requirements related to cybersecurity risk management for road vehicles that include electrical and electronic (E/E) systems. A joint working group of ISO and SAE experts develop ISO/SAE 21434. It will replace SAE J3061 and provides a framework, which supports the establishment of a cybersecurity culture during the complete product life cycle. Since cybersecurity risks can increase during the products lifetime it demands a management system, which is able to monitor changes in the threat landscape, vulnerabilities, etc. and provide updates from postproduction until decommissioning.

The standard recommends the definition of specific metrics for process rigour and risk level. In the presented approach the SecRL (Security Risk Level) has been defined to quantify risk and to perform variability management. In general, different security risk levels are needed for different attributes (privacy, operational, financial, safety). The paper at hand only deals with a risk level for functional safety. The standard is currently under development. The expected date of publishing is Q4 2020.

In addition, standards demand an established engineering process according to state of the art automotive quality standards (e.g. ASPICE, CMMI, IATF 16949).

2.2 Co-engineering life cycle for safety and security

In this subsection, we recall the method for co-engineering used in the core of this paper.

Security-informed Safety-oriented Process Line Engineering (SiSoPLE) [13] is a co-engineering method, which represents the extension of SoPLE, Safety-oriented Process Line Engineering [14], [15]. Similar to SoPLE, SiSoPLE consists of a two-phase method for engineering families of safety life cycles/processes. The first phase is aimed at engineering the domain from a process perspective i.e., identifying and systematizing process-related commonalities and variabilities, focusing on security-informed safety-related commonalities and variabilities, in order to concurrently engineer a set of processes. The second phase is aimed at deriving single processes via selection and composition of commonalities and variabilities. From a tooling perspective, SiSoPLE as well as SoPLE can be supported by the integration between EPF Composer, recently re-brought to life [20], and BVR Tool [31]. This integration was qualitatively evaluated as promising in [11] and its implementation was presented in [20]. To make the paper self-contained, we recall basic information regarding EPF Composer and BVR Tool.

EPF Composer implements a metamodel which exhibit a satisfactory overlapping with the SPEM (Software & Systems Process Engineering Metamodel) 2.0 language [30]. EPF Composer enables authoring, tailoring and deploying engineering life cycles and processes. This means that process structures containing all necessary process elements (e.g., activities, tasks, roles, work products, etc.) can be specified.

The BVR Tool implements the BVR (Base Variability Resolution) [18] language, built on top of CVL (Common Variability Language) [19] enable variability modelling in the context of the engineering of families of safety-critical systems. BVR enables orthogonal variability management for any model (called Base model), instance of a Meta-Object Facility (MOF)-compliant metamodel. Via the BVR Tool, variability engineers create three kinds of models:

VSpec models specify Feature-Oriented Domain Analysis (FODA) [22] -like models. To specify cross-branches constraints, which limit inclusion/exclusion within a subtree based on choices on other subtrees, Basic Constraint Language (BCL) is used.

Resolution models define the desired inclusion/exclusion choices for the specific configuration/resolution.

Realization models specify the placement fragments (i.e., sets of elements forming conceptual holes in a base model, which may be replaced by replacement fragments) and replacements within the fragment substitutions. A Fragment substitution is an operation that, if executed, substitutes a model fragment (placement fragment) with another (replacement fragment).

2.3 Safety and security co-analysis

In this subsection, we recall the method for co-analysis used in the core of this paper.

EVITA [4] is used to quantify the risk of potential cyberattacks. A risk level is derived based on "attack potential", "attack probability", "severity" and "controllability". It is a criterion that indicates the risk that functional safety can possibly be levered out by an attacker in certain circumstances.

Based on HARA in [24] SAHARA (Security-Aware Hazard Analysis and Risk Assessment) was introduced. It combines HARA from the safety and the STRIDE approach from the security domain. The intention of SAHARA is to identify security issues which can have an impact to safety concepts on system level. It also considers impacts which can occur because of safety issues.

FMVEA (Failure Mode, Vulnerabilities and Effect Analysis), [27], extends the FMEA and performs a combined safety and security analysis. It considers threat modes and failure modes. Threat modes describe possible ways how the security attribute of a component may fail caused by vulnerabilities. FMVEA determines the probability of a threat mode based on identified attack scenarios and vulnerabilities.

In [23] the relationship between HARA and TARA was investigated with regard to a joint assurance case.

3 Problem statement

Since vehicles provide highly interconnected system functions realized in software, the systems are no longer isolated. They become cyber-physical and cybersecurity has to be part of the centre of interest. Existing safety-related processes have to be expanded with methods like threat analysis and risk assessment and attack tree analysis.

An important aspect is the identification of relationships between cybersecurity and safety because freedom of interference has to be guaranteed. It is possible that security threats have impact to safety, if safety functions are implemented in software. Whereas safety deals with hazards and mishaps cybersecurity addresses threats resulting from malicious intent from external to the E/E system.

The methodology described in the next chapter is intended to identify all possible ways how functional safety may be violated in the different development lifecycle phases. In a combined process cybersecurity and safety risks will be identified jointly. In this context it has to be considered that there are risks which are only related to safety issues (e.g. hardware failure) and risks which are only related to cybersecurity (e.g. attackers want to capture personal data). Cybersecurity risks without safety relation will be possibly identified but they are out of scope from the perspective of the paper at hand.

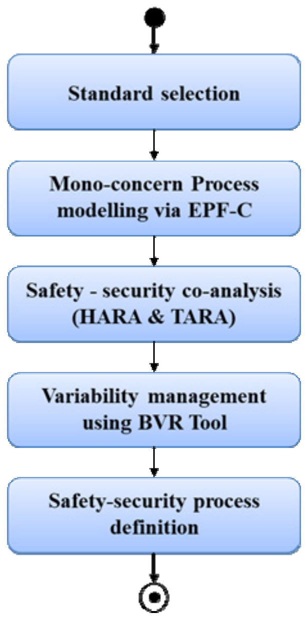
Based on analogies between safety and cybersecurity it is useful to define processes, which are integrating both topics. An integrated point of view is necessary because joint safety and security analysis will lead to measures, which have the objective to mitigate identified risks, which can be caused by both disciplines.

In the initial situation process developers have to work with standards which describe separated topics. Engineering teams in companies need an integrated development process which deals with quality, safety and security on different levels for different projects. Process developers have to harmonize several standards in their processes and provide evidence to all engineering areas.

Highly connected vehicles need a process to track the security status during the whole lifetime because previously unknown attacks may have the opportunity to compromise functional safety.

4 Methodology to define a process flow

To define a joint safety and security process, based on available but separated processes, it is necessary to have a systematic procedure to identify commonalities and variabilities. The proposed way is to use SiSoPLE, which is able to define joint processes. This chapter describes the development of two independent standard compliant processes for safety and security. These processes are the base for variants with project specific ASIL, SecRL and quality. The safety and security co-analysis delivers ASIL and SecRL, which are parameters for the variability management. The underlying workflow is illustrated in **¡Error! No se encuentra el origen de la referencia.**.


Standard selection
Mono-concern Process modelling via EPF-C
Safety - security co-analysis (HARA & TARA)
Variability management using BVR Tool
Safety-security process definition
Figure 1 Workflow for safety-security co-engineering process

Activities in cross concern applications, which have to be executed in any case, are called safety security co-engineering activities (instead of the single concern "commonality"). This term definition allows the extension of indicated activities, because it is the intention to "maximize" co-engineering activities and reduce variabilities if it is possible. Co-engineering capable methods can deal with both areas and they do not need to be a commonality in a strict sense. In this generalized view, it is sufficient that the methods head to the same goal.

4.1 Standard selection

The first step to create a process is to define which requirements it must fulfil. At least standards, which are demanded by legislator and customers, have to be considered.

4.2 Process modelling

The base process and the related model contain all activities, which can possibly be part of the development process and are directly related to the underlying standards. This means that all activities which may be needed for any ASIL and any SecRL are modelled. Later on, company specific activities and realisations will be added in the process definition step. Finally, for all concerns process models (mono concern models) are available. They are the basis for the following variability management.

4.3 Safety and security co-analysis

As recalled in Section 2.1, safety and security co-analysis is an important step in the concept phase, which has major impact to the following engineering activities. The described approach uses the resulting ASIL and SecRL as parameters to manage variability and define the co-engineering process.

Co-analysis in the concept phase has to make sure that interaction between different concerns is considered, because it should ensure that cybersecurity issues are considered as well as safety. The approach should guarantee that any additional potential hazards will be identified, which would stay undiscovered if only one discipline is examined in an isolated way. HARA and TARA must be performed in parallel but interweaved and consider potential dependencies between safety and security. The management of interaction between safety and security in an assessment is addressed in specific research papers (see section 2).

Identification of hazards and potential causes is an indispensable prerequisite for a safe and secure system. Hazards and threats from both areas need to be identified because unknown issues can lead to unsafe control actions, independent whether the cause is related to a hardware fault (classic safety-oriented view) or to a security issue. The goal is to define measures that are appropriate to mitigate any identified risks. To make sure that measures from competitive disciplines do not influence each other in a non-admissible way, a trade-off in the risk reduction measures has to be considered. The impact of each single safety and security measure needs an evaluation to find a balance.

Finally, arguments have to be collected in the assurance case, which covers the integrated and harmonized safety and security case, to show that the implemented measures are conform with underlying standards.

4.4 Variability management

Variability management is based on the defined ASIL and SecRL parameter set (see Figure 2). These two parameters have a major impact to the extent of the minimum required process activities. Tailoring of the base processes to a project specific multi-concern process means that unneeded activities are removed, and new project specific activities are added. Standards or company specific regulations

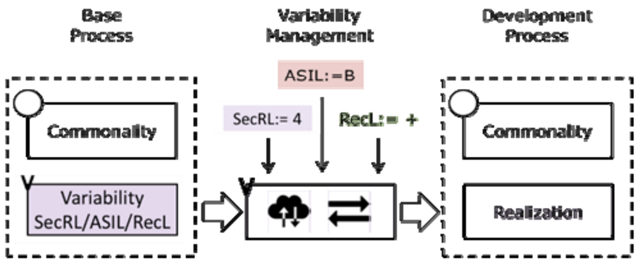


Figure 2 Variability management based on parameters [2]

demand for the application of a defined set of methods for a particular ASIL or SecRL. The development process must deal with variability because ASIL and SecRL varies for different items and in different projects.

BVR provides a mechanism to change activities and methods for various items according to different parameter sets. This feature is implemented by the usage of choices and constraints in the VSpec and the Resolution diagram (see Figure 3) of the BVR tool. The procedure how to build a process model and how BVR works in detail is described in the case study and in chapter "Management of families/lines" in Deliverable 6.3 [2].

An important feature that allows compliance checking is the verification function of the BVR Tool. This function uses constraints to evaluate the process model to make sure that it is compliant with the underlying standards. If the constraints are defined (this work is done only once) BVR can verify the model and all its alternative variants to identify modelling mistakes, which prevent a model from being standard compliant.

4.5 Definition of joint process

Process designers can use the parametrized model as starting point to integrate company and project specific requirements to get the joint process that implies demanded quality aspects and provides the wanted level of safety and security.

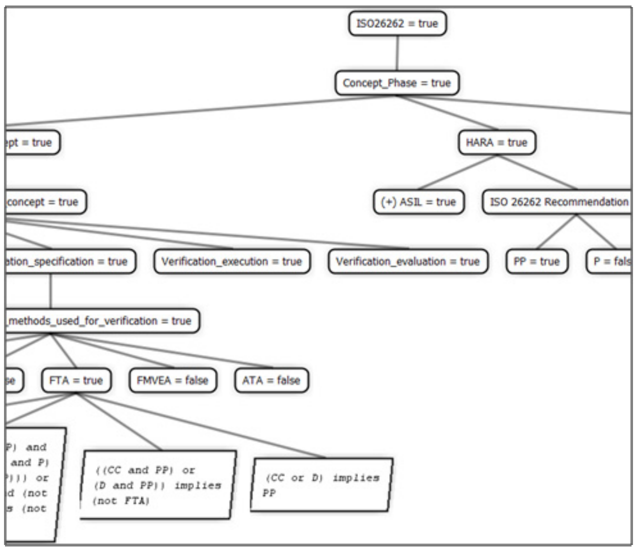


Figure 3 BVR Resolution diagram

5 Case study

In this section, we report about the case study. More specifically, we illustrate the application of our approach, focusing on the concept phase, to a collaborative security and safety-critical system.

5.1 System and scenario description

The case study uses a fleet of autonomous (model) cars that communicate at runtime via car2car communication to form a platoon (the interested reader may refer to the AMASS Deliverable 1.6 [2] for further details). The fleet constitutes a safety- and security-critical system of systems. The focus is on safety and security aspects of the radio connection, which is enabled by the car2car communication management unit. Precisely, the scenario in focus is as follows: an attacker threatens the fleet's integrity by adding unauthorised code to the communication manager unit. The execution of this code increases the CPU load to a forbidden level. As result the communication breaks down and the platooning function is not any longer available (hazardous event). In a real life scenario this hazardous event may cause harm to people.

Thus, in our scenario, the communication management unit loses its functionality (safety issue) triggered by a cybersecurity attack. Once the cybersecurity issue is identified, the software must be updated and also the hazard analysis needs a reverification to guarantee that it is still valid. Engineers have to check that there are no unwanted side-effects of the security update on any safety aspects.

5.2 Objectives

The objective of the case study is the definition and evaluation of a joint process for co-engineering safety and cybersecurity.

5.3 Application of process flow

For the cybersecurity- and safety-critical system under consideration, ISO 26262 and SAE J3061 are identified as relevant. Then, the process modelling in EPF-C begins. The obtained process model is a direct representation of the underlying standards and contains all addressed activities. It is called base model and is used to perform process tailoring, where unwanted activities are removed and new project specific ones are added.

According to the concept phase of ISO 26262, item definition and HARA needs to be performed. SAE J3061 demands a feature definition and a security analysis. In the system under consideration a combined hazard and threat analysis was performed with the tool ANSYS medini analyze [1]. The analysis was done using EVITA, which is one of the supported methods. The outcome was ASIL=B related to functional safety and SecRL=4 related to cybersecurity. ASIL B demands a minimal set of activities to achieve compliance with ISO 26262 and leads to safety measures to undercut the allowed failure rate.

Currently, standards do not define strict process requirements for (cyber)security, but it is demanded to have a defined process and a consistent line of arguments, when the product is brought to the market. Based on the standard compliant minimal set and the project specific requirements, the process variability management via the BVR Tool is started. As recalled in the background section, this requires the creation of three models: VSpec, Resolution, and Realization.

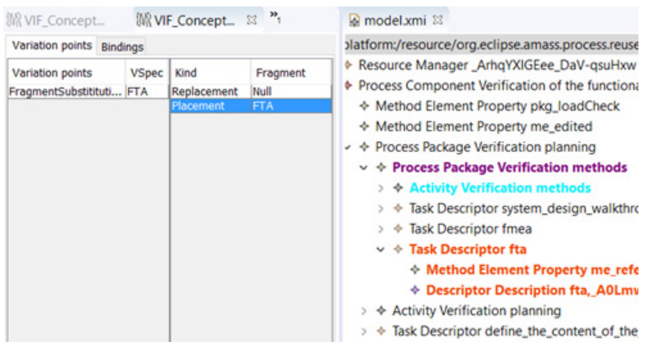
Our created VSpec model focuses on activities that vary in relation to ASIL and SecRL. Alternatives (XOR relation) and optionality (0/1) are also specified in the VSpec model. Once the VSpec is created, we can generate the Resolution model, as shown in Figure 3. Having set ASIL=B and SecRL=4, we are able to resolve the variability within our resolution model by choosing the appropriate features, where the variability parameters ASIL and SecRL decide whether process activities and specific methods have to be executed or not. More precisely, we assign "true" xor "false" to each activity of the model to define the process model, which will only include the features with true-value assignment and constraints satisfaction. Constraints make sure that all necessary activities are part of the model. They use logic operations to link elements. In the example shown in Figure 3 "(CC or D) implies PP" means that if ASIL C or D is selected also PP (++) has to be selected to receive a valid validation result. Once constraints are defined, they can be evaluated as often as needed if the BVR-function "Validate" is selected. If the validation is "true" the created model complies with the defined constraints and the requirements of the underlying standard.

Finally, we are ready to create the realization model, where the binding between the abstract representation of the desired/re-configured/resolved process, representing the joint process, and the concrete representation, expected to be rendered by EPF-C, is specified via a set of substitution rules.

In the realization model, partly shown in Figure 4, we specify that FTA, shown in Figure 3, shall be removed because it is a deductive analysis method, which is not highly recommended for ASIL B. Specifically, placement (FTA) and replacement (null) are specified.

Since FMEA is required for all ASILs, no substitution is included. Similar considerations are valid for FMVEA.

When the process and possible substitutions are executed/realized, the final process model is exported to an



Variation points	VSpec	Kind	Fragment
FragmentSubstituti...	FTA	Replacement	Null
		Placement	FTA

```
platform:/resource/org.eclipse.amass.process.reuse
Resource Manager _ArhqYXIGEee_DaV-qsuHxw
Process Component Verification of the functiona
  Method Element Property pkg_loadCheck
  Method Element Property me_edited
  Process Package Verification planning
    Process Package Verification methods
      Activity Verification methods
      Task Descriptor system_design_walkthro
      Task Descriptor fmea
      Task Descriptor fta
        Method Element Property me_refe
        Descriptor Description fta,_A0Lmw
    Activity Verification planning
    Task Descriptor define_the_content_of_the
```

Figure 4 BVR Realization diagram

EPF-C processable XMI format. The tailored and standard compliant model is now available in EPF-C. BVR Tool supports variability management and makes sure that all relevant activities and methods are part of the final model.

In our elaborated joint process model, FMEA is used in combination with FMVEA to perform a joint safety and security analysis to specify requirements for functional safety and cybersecurity.

5.4 Discussion and lessons learned

Besides safety issues the joint process has to cover security aspects as well. The behaviour of security issues is different to safety. From the safety point of view, it is sufficient to analyse the item and implement measures which make sure that the intended ASIL will be achieved. If no safety issue has been missed, developers can assume that implemented ASIL is valid without a time limit. From the security point of view the situation is different because malicious attacks have to be considered. In the example above, the communication management unit has lost its functionality (safety issue) triggered by a cybersecurity attack. It is also important to investigate the possibility that safety issues enable attackers to find new attack paths. SAHARA is a methodology which is able to support an analysis towards this direction.

The SecRL covers only the risk but it is not an adequate metric for process rigour and the related engineering effort. **Process rigour** might be a key indicator used as argument that during the engineering phase a sufficient combination of activities has been taken into account.

Metrics are not covered in normative parts of the security standards. Therefore, developers have to define some kind of process specific process rigour. Discussions to develop a framework concerning a cybersecurity assurance level is ongoing in standardisation working groups.

ISO 26262 demands processes to maintain functional safety during operation. Related to this requirement, a **field monitoring and update procedure** has to be available in the development process to ensure functional safety **until decommissioning**. Safety and especially security monitoring increase the effort, but it is absolutely essential with regard to automated driving functions.

Determination of risk in the early phases (e.g. TARA in concept phase) is based on parameters which **will change during the development** phase because "public" tools and methodologies to perform attacks will be improved and can influence parameters in the analysis. **Regular updates of the threat analysis** have to be planned (e.g. once a year) to check the validity of assumptions. A sole threat assessment before SOP is not sufficient in all cases because new attack paths may be developed.

In particular, the phase starting with post-production until decommissioning is very important for the security engineering life cycle. Cybersecurity monitoring during the use of items in the field will bring up information about new threats and vulnerabilities which are basis for a response plan.

6 Conclusion and future work

In this paper, we proposed a process for co-engineering safety and security by the explicit systematization and management of commonalities and variabilities, implicitly stated in the requirements of the different standards. Our process treats the safety and security life cycles as members of a security-informed safety-oriented process line. It forces safety and security engineers to come together and brainstorm on what might be considered a commonality and what might be considered a variability. We illustrated the usage of our process by systematizing commonalities and variabilities at risk analysis phase in the context of ISO 26262 for functional safety and SAE J3061 for cybersecurity. We obtained a SiSoPL from which we derived the intended process for co-engineering and our lesson learned. However, cybersecurity is an ongoing development towards ISO/SAE 21434, which will extend the process with new activities and steps. While functional safety is more stable there are also developments to extend the consideration from functional safety towards Safety Of The Intended Functionality (SOTIF) [7]. SOTIF describes a situation where hazards can be caused by insufficient performance or insufficient knowledge about the later environment. In a similar direction UL4600 [10] goes towards a guidance document regarding the evaluation of automated driving. A focus is here also on the reduction of unknowns, e.g. a process to generate understanding about the later environment and potential scenarios for complex systems. Recent examples and the new research on adversarial images [25] show that security is also an important consideration for safety of the intended functionality. For such systems, a life cycle targeting the co-engineering of safety and cybersecurity needs to consider the potential adversarial impact on the environment of an automated system. Thus, the extension of our SiSoPL, considering the interplay of the different and relevant standards and guidance within the automotive domain, constitutes part of our future work. We also aim at quantitatively evaluating the tailoring enabled by our SiSoPL, as done in [17], within the space domain.

Acknowledgment. This work is supported by the EU projects AMASS and Secredas [9]. Research leading to these results has received funding from the EU ECSEL Joint Undertaking under grant agreement n° 692474 (project AMASS) and n°783119 (project Secredas) and from Sweden's Vinnova, and Sweden's Knowledge Foundation via the SACSys (Safe and Secure Adaptive Collaborative Systems) project, the COMET K2 - Competence Centres for Excellent Technologies Programme of the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit), the Austrian Federal Ministry of Science, Research and Economy (bmwfw), the Austrian Research Promotion Agency (FFG), the Province of Styria, and the Styrian Business Promotion Agency (SFG).
ANSYS supported the AMASS project with academic licences for their commercial tool medini analyze.

References

[1] AMASS (Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems) Project (online), https://www.amass-ecsel.eu.

[2] AMASS Project: Deliverables (online) https://www.amass-ecsel.eu/content/deliverables.

[3] ENISA (European Network and Information Security Agency), *ENISA good practices for security of Smart Cars*, https://www.enisa.europa.eu/publications/enisa-good-practices-for-security-of-smart-cars.

[4] EVITA project, https://www.evita-project.org.

[5] ISO 26262 (2018), *Road vehicles – Functional safety,* International Standard.

[6] ISO/SAE 21434, *Road vehicles – Cybersecurity Engineering - General Overview.* https://www.iso.org/standard/70918.html

[7] ISO/PAS 21448 (2019), *Road vehicles - Safety of the intended functionality*.

[8] SAE - Society of Automotive Engineers, *SAE J3061 - Cybersecurity Guidebook for Cyber-Physical Automotive Systems*.

[9] SECREDAS (Product Security for Cross Domain Reliable Dependable Automated Systems), http://secredas.eu/

[10] Underwriters Laboratories Inc. (UL), *UL 4600 - Standard for Safety for the Evaluation of Autonomous Products*.

[11] I. Ayala, B. Gallina (2016), *Towards Tool-based Security-informed Safety Oriented Process Line Engineering,* 1st ACM International workshop on Interplay of Security, Safety and System/Software Architecture (ISSA), Copenhagen, Denmark.

[12] J. Castellanos Ardila, B. Gallina (2017), *Towards Efficiently Checking Compliance Against Automotive Security and Safety Standards*, 7th IEEE International Workshop on Software Certification., Toulouse, France.

[13] B. Gallina, L. Fabre (2015), *Benefits of security-informed safety-oriented process line engineering*, Digital Avionics Systems Conference (DASC), IEEE/AIAA 34th (pp. 8C1-1), IEEE.

[14] B. Gallina, I. Sljivo, O. Jaradat (2012), *Towards a Safety-oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification,* Post-proceedings of the 35th IEEE Software Engineering Workshop (SEW-35).

[15] B. Gallina, S. Kashiyarandi, H. Martin, R. Bramberger (2014), *Modeling a safety-and automotive-oriented process line to enable reuse and flexible process derivation*, Computer Software and Applications

Conference Workshops (COMPSACW), IEEE 38th International, pp. 504-509.

[16] B. Gallina, M. A. Javed, H. Martin, R. Bramberger (2019), *Co-engineering of security and safety life-cycles for engineering security-informed safety-critical automotive systems in compliance with SAE J3061 and ISO 26262*, 24th International Conference on Reliable Software Technologies-Industrial Presentation Track (Ada-Europe), Warsaw, Poland, June 11-14.

[17] B. Gallina (2019), *Quantitative Evaluation of Tailoring within SPICE-compliant Security-informed Safety-oriented Process Lines*, Journal of Software: Evolution and Process, EuroSPI Special Issue, DOI:10.1002/smr.2212.

[18] Ø. Haugen, O. Øgård (2014), *BVR–better variability results*, International Conference on System Analysis and Modeling (pp. 1-15). Springer, Cham.

[19] Ø. Haugen (2012), *Common Variability Language (CVL),* Object Management Group, Tech. Rep. ad/2012-08-05 [Online]. Available: http://www.omgwiki.org/variability/doku.php

[20] M. A. Javed, B. Gallina (2018), *Get EPF Composer back to the future: A trip from Galileo to Photon after 11 years*, EclipseCon, Toulouse, France.

[21] M. A. Javed, B. Gallina (2018), *Safety-oriented process line engineering via seamless integration between EPF composer and BVR tool*, Proceedings of the 22nd International Conference on Systems and Software Product Line-Volume 2 (pp. 23-28), ACM.

[22] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson (1990*), Feature-oriented domain analysis (FODA) feasibility study (No. CMU/SEI-90-TR-21),* Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

[23] H. Martin, R. Bramberger, C. Schmittner, Z. Ma, T. Gruber, A. Ruiz, G. Macher (2017), *Safety and security co-engineering and argumentation framework*, International Conference on Computer Safety, Reliability, and Security (pp. 286-297). Springer, Cham.

[24] G. Macher, E. Armengaud, C. Kreiner, E. Brenner, C. Schmittner, Z. Ma, M. Krammer (2018), *Integration of security in the development lifecycle of dependable automotive CPS,* Solutions for Cyber-Physical Systems Ubiquity (pp. 383-423), IGI Global.

[25] N. Morgulis, A. Kreines, S. Mendelowitz, Y. Weisglass (2019), *Fooling a Real Car with Adversarial Traffic Signs*, arXiv preprint arXiv:1907.00374.

[26] A. Ruiz, B. Gallina, J. L. de la Vara, S. Mazzini, H. Espinoza (2016), *Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems*, 5th International Workshop on Next Generation of System Assurance Approaches for Safety-Critical Systems (SASSUR), Trondheim.

[27] C. Schmittner, T. Gruber, P. Puschner, E. Schoitsch (2014), *Security application of failure mode and effect analysis (FMEA),* International Conference on Computer Safety, Reliability, and Security (pp. 310-325), Springer, Cham.

[28] J. L. de la Vara, E. Parra Corredor, A. Ruiz Lopez, B. Gallina (2019), *AMASS: A Large-Scale European Project to Improve the Assurance and Certification of Cyber-Physical Systems*, Proceedings of the 20th International Conference on Product-Focused Software Process Improvement (PROFES), Barcelona, Spain.

[29] J. L. de la Vara, A. Ruiz, B. Gallina, G. Blondelle, E. Alaña, H. Herrero, F. Warg, M. Skoglund, R. Bramberger (2019), *The AMASS Approach for Assurance and Certification of Critical Systems,* embedded world Conference (ewC), Nuremberg, Germany.

[30] OMG (2008), *Software & systems Process Engineering Meta-model (SPEM), v 2.0*, Full Specification formal/08-04-01.

[31] BVR Tool. https://github.com/SINTEF-9012/bvr

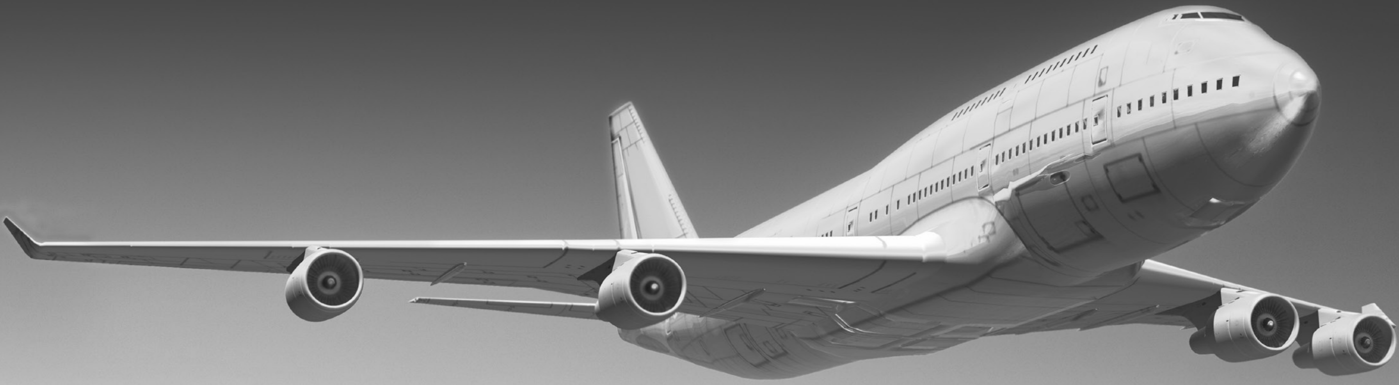
[32] OpenCert - hosting the AMASS platform. https://www.polarsys.org/opencert/about/

[33] Eclipse Process Framework, Eclipse Foundation, Inc., Canada, http://www.eclipse.org/epf/.

[34] ANSYS medini analyse, ANSYS Inc., USA, https://www.ansys.com/products/systems/ansys-medini-analyze

Revolutionize your software verification

Efficiency, Automation, Reliability



RAPITA Systems
A DANLAW Company

Unit testing · System testing · Coverage analysis · Timing analysis
V&V services · Multicore timing services · DO-178C training
Ada · C · C++

www.rapitasystems.com

Using Evidence-Based Arguments to Support Dependability Assurance – Experiences and Challenges

Janusz Górski

Gdańsk University of Technology, Narutowicza 11/12, 80-233 Gdańsk, Poland; Tel: +48 58 347 1909; email: jango@pg.edu.pl

Abstract

The presentation introduces to the problem of evidence-based arguments and their applications. Then, based on the experiences collected during development and commercial deployment of a concrete solution to this problem (system NOR-STA) we overview selected challenges and the ways of addressing them.

Keywords: Evidence-based argument, assurance case, conformance case, tool support.

1 Introduction

The interest in using explicit evidence-based arguments with respect to socio-technical systems was growing over last forty years. It originated from the concept of *safety case* addressing the need to demonstrate safety and then was generalized to the concept of *assurance case* addressing a broader scope of objectives (like security, reliability, privacy). It has been also recognized that explicit evidence-based arguments can be used to demonstrate conformity with pre-defined sets of structured requirements of standards and other normative documents, which resulted in the concept of *conformance case.*

In this paper we briefly describe selected challenges which we were facing while developing, implementing and deploying the Trust-IT methodology and the NOR-STA system supporting applications of evidence-based arguments. NOR-STA has been gradually developed in a series R&D projects: EU sponsored projects DRIVE, PIPS and ANGEL, Polish-Norwegian Research Fund sponsored project ERM and European Regional Development Fund sponsored project NOR-STA. Since 2014 NOR-STA is a commercial product offered by Argevide, a spin-off company of Gdansk University of Technology [1]. More about challenges and the related solutions implemented in NOR-STA can be found in [2].

2 About evidence- based arguments

Argument is an attempt to persuade someone of something, by giving reasons and/or evidence for accepting a particular conclusion**.** This 'something' we want to argue about may be, for instance, assurance of some important property (like safety, security, privacy, reliability), conformance with a stated set of criteria (imposed by a standard, norm, directive, recommendation) or any other property selected as being of interest to the parties exchanging the arguments. An example argument could be:

> ***Module correctness argument:***
>
> *Tests confirm that this software module meets its requirements because test results are positive and the tests coverage is sufficient.*

Looking more closely to this argument we can identify two parts which are of different nature: the *logic part* and the *epistemic part.*

The logic part establishes the 'conveyance' relationship between the conclusion (also called *claim*) of the argument and the premises of the argument. In our example the claim postulates that '*the module meets requirements*' and the premise postulates the fact: '*test results are positive and the test coverage is sufficient'*. The 'conveyance' relationship between the two is established by the *strategy of argumentation* (the inference rule) which asserts that from the truth of the premise we can conclude the claim. It usually needs some *rationale* justifying the reasons for acceptance of such strategy. In our example the strategy of argumentation is: *argumentation by referring to test results and test coverage* and the rationale could be: *experience shows that positive results of tests of adequate coverage reliably demonstrate fulfilment of the requirements*. A graphical representation of the logic part of our example argument is given in Figure 1.

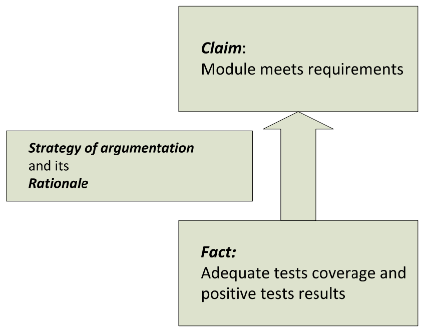

Claim:
Module meets requirements
Strategy of argumentation and its Rationale
Fact:
Adequate tests coverage and positive tests results

Figure 1 Logic part of *Module correctness argument*

The epistemic part of the argument focuses on providing *evidence* which in its broadest sense includes everything that can be used to determine or demonstrate the truth of the fact referred to in the argument. For instance, the fact *it is raining outside* could be demonstrated by a video stream from the camera looking outside through the window.

In our *Module correctness argument* example, such evidence could include the test plan and the test results for the considered software module. In our further considerations we assume that evidence is delivered in electronic form: text, graphics, image, video, audio, sensor measurements, etc. A graphical model of the epistemic part of the example argument is given in Figure 2.

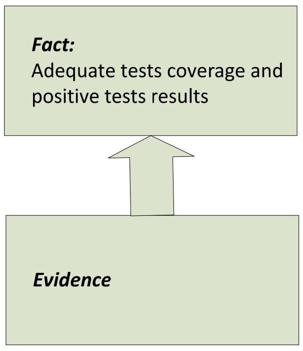


Figure 2 Epistemic part of *Module correctness argument*

In general, the premises of an argument can, in addition to facts, also include *assumptions* imposing constraints on the context of argumentation as well as more specific claims (*sub-claims*) which need further argumentation. This latter possibility results in hierarchical argumentation

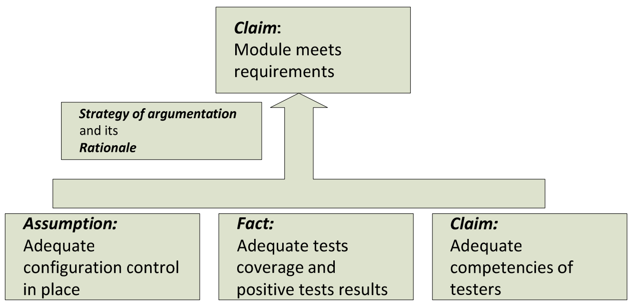

Claim: Module meets requirements
Strategy of argumentation and its Rationale
Assumption: Adequate configuration control in place
Fact: Adequate tests coverage and positive tests results
Claim: Adequate competencies of testers

Figure 3 Extended *Module correctness argument*

structures of an arbitrary depth. Possible extension of the *Module correctness argument* by introducing additional premises is illustrated in Figure 3.

Convincing arguments can be used to build trust, because they demonstrate trustworthiness. Such arguments we call *trust cases.* For example, a convincing (supported by evidence) argument that *a service is secure* increases trust in the service. The evidence supporting such argument could include: protective security measures used, certification procedures passed, penetration tests results, operating data, development practices used and so on.

In such case, the strategy of argumentation is modified to: *argumentation by referring to test results, test coverage and testers' competencies with the assumption that adequate configuration control is in place.*

In our research we have particular interest in two different types of trust cases: *assurance cases* which focus on demonstrating assurance of some chosen (and considered important) property (like safety, security, privacy, dependability, reliability etc.) and *conformance cases* where the focus is on demonstrating conformity with some predefined set of requirements (given in standards, norms, directives, regulations etc.).

The primary objects of interest for developing trust cases are ICT products, services and processes, however the scope of applicability of trust cases it very broad and includes all situations where human or technical objects establish trust relationships by exchanging arguments demonstrating their mutual trustworthiness.

3 Argument representation

The main challenge is to decide about the argument model and the corresponding language of expressing arguments to provide for adequate expressive power, understandability and scalability of arguments.

The model used in NOR-STA is presented in Figure 4.

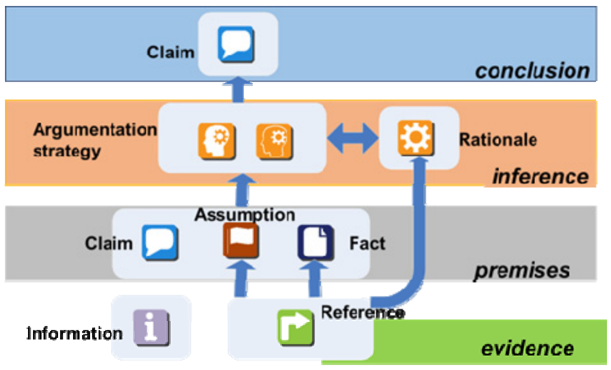
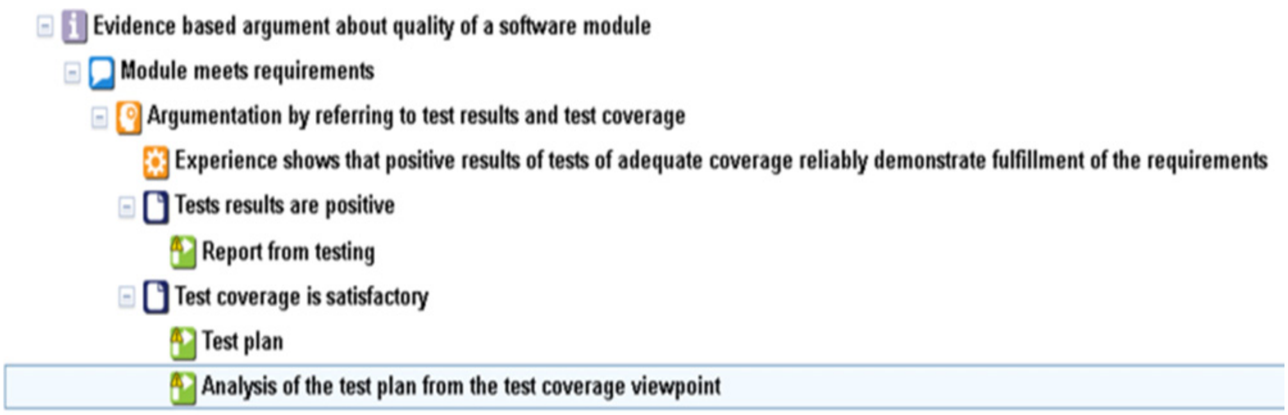


Figure 4 Argument model in NOR-STA

According to the model of Figure 4, the nodes (elements) of an argument are represented by different graphical icons. The icons can have textual descriptions (fitting to a single line) and in addition can have richer descriptions accessible after selecting a given node. The hierarchy of argumentation develops from left to right, as a set of structured lines, each line marked by a proper icon. For instance, the example *Module correctness argument* can be, following the model of Figure 4, represented as in Figure 5.

4 Communication and co-operation

To fulfil their role of supporting building and establishing trust, arguments need to be easily communicated between the interested parties. This leads to the requirement of controlled argument sharing with the objective to provide easy access by the authorized parties and simultaneously



Evidence based argument about quality of a software module
Module meets requirements
Argumentation by referring to test results and test coverage
Experience shows that positive results of tests of adequate coverage reliably demonstrate fulfillment of the requirements
Tests results are positive
Report from testing
Test coverage is satisfactory
Test plan
Analysis of the test plan from the test coverage viewpoint

Figure 5 ***Module correctness argument*** **in NOR-STA**

to provide adequate protection against unauthorized accesses. Different roles can be identified while accessing an argument, for instance argument developer, argument assessor, argument viewer, argument administrator and so on. Each of these roles can be refined according to the needs, for instance we can distinguish different sub-roles of argument developer: those responsible for logic part of the argument and those responsible for the epistemic part (suppliers of evidence). Different roles may also be associated with different views at the argument, for instance an auditor of a conformance case can see the standard requirements and the associated evidence in a form which best supports his/her task of assessing conformance with the standard.

The above considerations led to key decisions related to the NOR-STA system:

- Deploying NOR-STA in accordance with the SaaS (Software-as-a-Service) model.
- Managing access control in accordance with the RBAC (Role-Based Access Control) model.
- Providing different views to support different roles the users play with respect to a given argument.

5 Argument assessment

Argument assessment is necessary in different scenarios, like decision making, consensus building or disputes resolution.

Both, logic and epistemic parts of an argument are subjected to assessment. The assessment involves appraisal of the 'compelling power' of an argument. The assessment results can be selected from a two-value scale {*accept*, *reject*} like in case of a mathematical proof, or from a more complex space distinguishing different levels of acceptance/rejection and the related uncertainty. Consequently, we can have different *argument assessment mechanisms* which can be applied with respect to the same argumentation structure.

Referring to our *Module correctness argument*, the logic related question is: *do successful tests of right coverage really demonstrate that the module meets its requirements?*. And the epistemic question is: *do we really have positive test results and do the tests adequately cover the requirements?*

Answering positively to the logic question we confirm that meeting the requirements by a software module can be demonstrated by developing an adequate test plan, running the corresponding tests and receiving positive results of the tests. Note that if accepted, such argumentation strategy can be reused with respect to other software modules as well.

If we still doubt about the answer to the logic question, we can modify the argumentation strategy by adding additional premises. For instance, in case of our example argument, the additional premises could be: ***Assumption:*** *adequate configuration management in place* and ***Claim:*** *adequate competencies of testers* (as shown in Figure 3).

Answering positively to the epistemic question means that satisfactory evidence has been provided demonstrating that the assertion (represented by a given fact) is true in the considered context. For instance, the fact: *test results are positive* can be demonstrated by providing the report from tests whereas the assertion *test coverage is adequate* can be demonstrated by providing the test plan and the result of the analysis of this plan against the relevant set of requirements.

Depending on the applied assessment mechanism, the results of assessment are selected from different scales. An example of an advanced assessment mechanism can be the mechanism based on Dempster-Shafer belief functions implemented in NOR-STA, which supports the two dimensional space: ***Decision***={*rejectable*, *opposable*, *tolerable*, *acceptable*} and ***Confidence***={*sure*, *very high*, *high*, *low*, *very low*, *uncertain*} from which the assessor selects his/her assessments. The details of this assessment mechanism can be found in [3].

From the experience we have so far with the NOR-STA system, different application domains may require different argumentation assessment mechanisms and therefore it is essential that the tools supporting application of evidence-based argumentation were able to switch between different mechanisms depending on a particular usage context. Presently, the NOR-STA system implements some nine different argumentation assessment mechanisms and its architecture is open to easily absorb the new ones, if needed.

Let us consider a task of assessing a complex argument (multiple levels of the argumentation hierarchy and a large number of facts supported by related evidence). In most cases assessment of the epistemic part can be split into a number of independent (local) assessments: each fact can be considered in isolation and the assessor assesses to which extent the submitted evidence supports this fact (for instance, to which extent the submitted report from testing demonstrates that the results of tests are positive). The assessment of the logic part can also be localized, i.e. each argumentation strategy can be assessed in isolation by looking at its conclusion and its premises. The problem becomes more complex if we try to propagate (local) assessments of facts towards the assessments of claims which depend on these facts. In case of more advanced assessment mechanisms, manual realisation of this task can be very laborious and error prone. The solution is to define the aggregation rules which can then be implemented and performed automatically. For instance, such rules for the Dempster-Shafer based mechanism implemented in NOR-STA are documented in [3]. Having the aggregation of local assessments automated, we can assess 'large' arguments with a reasonable effort (NOR-STA users have such experience with arguments up to several thousands of nodes).

Another important issue is the way of presenting the argument assessment results. As arguments are (mostly) exchanged between people, it is of particular importance that the assessment results are presented in a human-friendly way and that they support tasks performed by the users. In our experience, using colours to distinguish different values from the argument assessment scale proven to be particularly effective (basic colours: red meaning rejection, green meaning acceptance and yellow meaning uncertainty). These basic colours can be then mixed to distinguish more fine values, while using more advanced assessment mechanisms. This way of visualization not only communicates the overall assessment (the assessment of the top claim of the argument) but also provides for easy identification of the 'weak' parts of the argumentation and supports decisions concerning improvement of the considered argument.

6 Size and change management

Arguments can be complex structures composed of a large number of nodes and integrating large number of pieces of evidence. Argument can also have a long lifespan during which the argument is subjected to changes and modifications. For instance, consider a conformance argument demonstrating that a given organisation is conformant with ISO27001 or a safety argument related to an autonomous vehicle. The scope of changes will include both, the structure and the evidence and the arguments will be subjected to different assessments (for instance, self-assessment, third party assessment, repeated assessment after certificate expiration and so on).

6.1 Operating 'large' arguments

Large arguments are difficult to handle and to understand (what does it mean 'large'? From NOR-STA users we have reports about arguments up to 8000 nodes). Representing hierarchies of this size in a graphical form causes problems with visualizing the hierarchy within the limits of a computer screen, inserting textual descriptions in graphical symbols and showing dependencies between nodes in a readable way. The NOR-STA way of representing the hierarchy from left-to-right (instead of from top-to-bottom) and representing each node in a single line is advantageous for large arguments (an analogy can be the commonly accepted way of presenting file directory structure as the left-to-right hierarchy instead of presenting it as a vertical graphical structure).

6.2 Managing massive evidence

A realistic argument (for instance, demonstrating conformance with a selected standard or demonstrating safety of a new technology to be applied off-shore) will integrate many different documents which contain evidence supporting the argumentation. These can be electronic documents of any format (textual, graphics, video, audio etc.) and the documents can reside in different locations with different access protocols (web pages, ftp, svn and so on). It is necessary to access these documents either in their target repositories or, alternatively, to provide for dedicated and adequately protected customized repositories. In many cases, the documents can be large (for instance, a design documentation of a medical device) and the evidence we want to refer to is a selected part of such document. In such case it is advantageous to have a possibility to refer to this particular part instead of referencing the whole document.

Often the evidence referred to in the argumentation is subjected to stringent security constraints (examples are personal data, trade secrets, reputation related data and so on). Therefore, while providing support for evidence-based arguments it is necessary to implement and to demonstrate conformance with (sometimes very demanding) security objectives which need to be met and continuously maintained.

6.3 Change management

Argument structure, the supporting evidence and argument assessments can be subjected to changes and modifications. This results in a continuous evolution of the whole argumentation and calls for an adequate change management mechanisms.

At the NOR-STA tool level, the following mechanisms proven their usefulness.

Accountability of changes where each modification introduced to the argumentation structure and to the assessments is recorded in the argument history providing for the identification of the responsible user.

Baseline mechanism where baseline is a (named) 'snapshot' of the current state of the whole argument. Such baseline can be later used as a well-defined reference (for instance, a full contents of the conformance argument which has been third-party assessed to obtain a formal certificate).

Rollback mechanism which provides access to the full history of changes and enables to roll-back to any previous moment from the history, if necessary (for instance, to choose an alternative way of developing the argument or to recover from a disaster).

7 Fitting into user business context

Assurance and conformance arguments have multiple stakeholders and it is important that these stakeholders can access the argument with their corresponding access rights. Therefore managing different user accounts and user access rights is necessary and the role of administrator of these accounts needs to be distinguished.

The users may maintain multiple arguments where each argument can have a different concern (for instance, conformance cases related to different standards, assurance arguments related to different products, arguments related to different objectives and so on). Therefore, it is necessary to provide for different 'working spaces' of different arguments and to support grouping arguments according to different criteria (for instance, different products, different standards, different assurance objectives, different organizational departments and so on).

The NOR-STA system supports users and user rights management and provides means for introducing structure into the set of different arguments. Each argument is maintained in its *project* (a sort of 'working space') where it undergoes its evolution. The projects can be arbitrarily organized into *folders* and the folders structure is hierarchical (resembling the file directory structure of operating systems). This mechanism provides for sufficient flexibility of organizing different arguments into a structure which meets expectations of user organisation. Purposeful grouping of projects into folders also helps in enforcing common access policies with respect to 'similar' arguments.

Assurance (or conformance) case can be treated as an electronic document which maintains a convincing argumentation that the user organization achieves some important objectives (for instance, meeting the safety requirements by its product or being conformant with selected standards). In the present business contexts however, it is often required that such information is presented in a more 'traditional' form, for instance as printed documents of predefined structure. Therefore the issue of reports generation cannot be neglected.

The solution applied in NOR-STA is to provide for a number of pre-defined reports of the metrics related to an argument and reports presenting the contents of the whole argument (in pdf and in doc formats) and in addition to this to provide for integrating with commonly accessible tools, like Excel, which support different forms of data presentation. Integration through XML/HTML data and XLS scripts to process XML data provide for high flexibility in generating reports in different structures and formats.

Figure 6 presents an example report generated from NOR-STA, where the *Module correctness argument* (shown in Figure 5) is presented in the GSN notation [4].

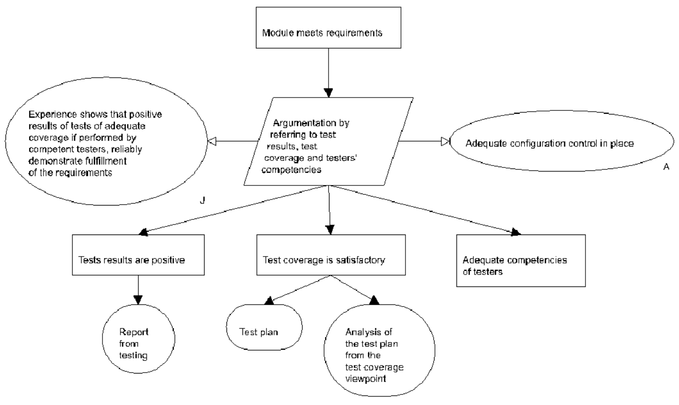

Module meets requirements
Experience shows that positive results of tests of adequate coverage if performed by competent testers, reliably demonstrate fulfillment of the requirements
J
Argumentation by referring to test results, test coverage and testers' competencies
Adequate configuration control in place
A
Tests results are positive
Test coverage is satisfactory
Adequate competencies of testers
Report from testing
Test plan
Analysis of the test plan from the test coverage viewpoint

Figure 6 GSN representation of *Module correctness argument*

8 Integration

Evidence-based argument is not being used in isolation. Instead, it has to be integrated within the broader context to which the argument is expected to bring an added value. In particular, this context can include other systems supporting the users' tasks and various repositories which store documents that are vital to achieving business objectives of the user organisation. These documents, produced by business processes (Design, V&V, QA, HR and others) are the sources of evidence which is referred to by the argument demonstrating achievement of the assurance/conformance goals of the organisation.

In NOR-STA system, the key to integration with other systems is the API (Application Program Interface) implemented as a set of web services which cover full functionality of NOR-STA. This provides a technical base for integration with selected external systems or services, according with the needs. Examples are Single Sign On (SSO), Active Directory Federation Services (ADFS), Azure B2C or Siemens Teamcenter.

Another technical base for integration is XML based export/import of the whole argument which can then be processed by dedicated applications, if needed. In particular, this provides for argument contents exchangeability with other tools supporting evidence-based argumentation.

9 Argument structuring and reuse

Evidence-based arguments can be structured following different (not necessarily orthogonal) decomposition criteria. Examples are: risks based decomposition where the argument addresses relevant risks and demonstrates that they are adequately mitigated or architecture/design model based decomposition where the argument follows the structure of the considered system, its subsystems and modules and demonstrates their selected properties.

More support is needed for automatic derivation of assurance case structure from the results of (standardized) risk analyses [5] or from the architectural/design models of a system [6].

Argumentation reuse has the potential of significant reduction of development effort by standardization of typical substructures recurring in arguments. A NOR-STA represented inventory of *design patterns* of arguments can be found in [7]. A particular pattern supporting the reuse of conformance cases is called *conformance template* [8]. This is the logic part of the argument which reflects the structure of the requirements of a selected standard. As long as the corresponding standard remain unchanged this logic part can be reused in in each conformance case which demonstrates conformity with the standard.

If the standard changes, however (and all 'living' standards undergo evolution), the changes need to be reflected in the conformance template and then propagated to all conformance arguments that were created following the template. NOR-STA supports such automatic propagation of changes introduced to a conformance template. The intention is to maintain consistency between the (changing) standard and its (multiple) applications in various target contexts.

10 Composability

Assurance/conformance cases are being used in different contexts. For instance, a component produced by its manufacturer is being delivered with its assurance case and after being sold to different buyers, is used in different systems. The developer of the assurance case of the target system would be interested to refer to the component assurance case and to reuse its assessment results. The questions arising in such scenario include: how to interface the component related assurance case to the system related assurance case, how to pass the assessment results of the component related assurance case and what if this result is context dependent (the assessment of the component related assurance case can be different depending on the target system context) and so on.

Presently, in NOR-STA system there are two mechanisms supporting composability of assurance/conformance cases: explicit representation of assumptions and *required/provided interfaces*.

Distinguishing a separate node type for representing assumptions (see Figure 4) provides for explicit enumeration of the assumptions conditioning a given evidence-based argument and protects against assumptions overlooking and omissions. In Figure 3 we have an explicit assumption that the module is being tested assuming that *adequate configuration control is in place* which prevents against situations where, for instance, the tests were performed according to an invalid test plan. While using the *Module correctness argument* within the context of the system embedding the module, we can verify if this assumption is still valid before accepting the result of the assessment of this argument.

NOR-STA also supports explicit declarations of interfaces between assurance/conformance case components. Consider an extended version of the assurance case of our example software module shown in Figure 3. The premise *adequate competencies of testers* is a claim which needs to be further demonstrated. Assume that this claim has been demonstrated by a separate *Tester competencies argument* which by declaring its *provided interface* make this claim and its assessment visible to the outside world. Inside the module, the claim is demonstrated by, for instance, using CV-s of the testers as the supporting evidence.

If the argument shown in Figure 3 declares as its *required interface* the claim *adequate competencies of testers* and if the two interfaces (the provided one and the required one) are *bound* together, then the two modules (the *Module correctness argument* and the *Tester competencies argument*) form a single argument independently of if the *Tester competencies argument* is also used in other contexts. The results of the assessment of the tester competencies will be automatically propagated to each assurance case which is bound with the *Tester competencies argument* through the provided/required interfaces.

11 Conclusions

Argument is a focal point situated between different stakeholders and addressing their concerns. By exchanging arguments the users can develop mutual trust that their concerns are being addressed with the satisfactory assurance.

In this presentation we have briefly characterized some of the main challenges and the related decisions which were made during development and deployment of NOR-STA – a system supporting development, assessment and maintenance of evidence-based arguments in different application contexts.

Presently, NOR-STA is used commercially in different domains, including medical, oil and gas, automotive, flight control and others. The short-term strategy of further development is customer-driven and follows the needs of current and prospective users. Equally important is also the long-term strategy which looks into the trends and tries to identify the future challenges, even if not yet articulated by the present customers. Two challenges can be considered as examples.

Firstly, better support for composability of arguments, not only at the syntactic level (provided/required interfaces) but also at the semantic level (matching the contexts within which arguments maintain their validity).

Secondly, continuous assessment of an argument which follows the changes in the evidence and automatically reflects these changes in the assessment of the argument. This could for instance support the concept of *continuous certification* as opposed to the present practices of repeated certification based on a predefined schedule (which is being criticized as inadequate for, for instance, security certificates in a very dynamically changing landscape of security threats).

Acknowledgement

Research, development and commercial deployment of NOR-STA had multiple contributors. In particular, the contribution of the following colleagues is to be acknowledged (in alphabetic order): dr Łukasz Cyra, Jakub Czyżnikiewicz (programmer), dr Aleksander Jarzębowicz, dr Jakub Miler, dr Andrzej Wardziński, Michał Witkowicz (programmer).

References

[1] Arevide sp. z o.o., www.argevide.com

[2] *Challenges in providing support for management of evidence based arguments*, https://www.argevide.com/wp-content/uploads/2016/05/Argevide-WP3-Challenges.pdf

[3] L. Cyra, and J. Górski (2011), *Support for Argument Structures Review and Assessment*, Reliability Engineering and System Safety (96), Elsevier, pp. 26-37.

[4] Goal Structuring Notation Community Standard, Version 2 (2018), https://scsc.uk/r141B:1?t=1

[5] A Wardzinski and P. Jones (2017), *Uniform Model Interface for Assurance Case Integration with System Models*, Computer Safety, Reliability, and Security, Springer, pp. 39-51.

[6] R Hawkins, I. Habli, D. Kolovos, R Paige, T. Kelly (2015), *Weaving an Assurance Case from Design: A Model-Based Approach*, IEEE Xplore.

[7] M. Szczygielska and A. Jarzębowicz (2018), *Assurance Case Patterns On-line Catalogue*, In: Advances in Dependability Engineering of Complex Systems. Springer, IND 141625.

Using SPARK to Ensure System to Software Integrity: A Case Study

T. Naks
AdaCore Estonia, Maealuse 2/1, 12618 Tallinn, Estonia;
Tallinn University of Technology; email: naks@adacore.com

M. A. Aiello, S. T. Taft
AdaCore Inc, 150 W. 30th Street, New York, NY 10001, USA; email: {aiello, taft}@adacore.com

Abstract

This paper describes work in progress on a workflow that supports consistent property-preservation proofs from early stages of system requirement specifications down to software requirements and final implementation. This workflow, called System-to-Software Integrity (SSI), demonstrates that the implemented software satisfies constraints defined in system requirements. In this paper, we demonstrate two important building blocks of this workflow. First, SysML to Simulink translation, which translates system level property specifications to Simulink, where it is easy to perform design of the control software. Second, Simulink to SPARK translation, which supports both checking the consistency of system-level property specifications as well as verification of property preservation in the software design.

The approach is demonstrated on a simple example of a car cruise control simulator.

Keywords: SPARK, SysML, Simulink, System-to-Software Integrity, observer, property preservation.

1 Introduction

Specification languages that incorporate contracts such as preconditions, postconditions, and invariants have been in existence for many years. Examples include the Z notation from Jean-Raymond Abrial [1], TLA and TLA+ from Leslie Lamport [2] and Alloy from Daniel Jackson [3]. Certain programming languages, programming language extensions, or software-development refinement toolsets have incorporated such contracts directly into their syntax, including Eiffel [4], SPARK (an Ada subset) [5], Ada 2012[1], Frama-C (C + ACSL) [6], and the Java Modeling Language (JML)[2]. In other cases, specification languages have been intended to be stand-alone specifications, with tools designed to help verify desirable properties, such as functional correctness, safety, or security at a level above the programming level [2,3]. SpeAR (Specification and Analysis of Requirements) is such a system based on "past" LTL [7], and AADL with AGREE and Resolute is another [8]. Model-level verification is generally based on compositional Assume/Guarantee reasoning, using a model checker [7,8,9].

[1]https://www.iso.org/standard/61507.html
[2]https://www.openjml.org

Relatively few systems have been developed where specifications are carried across levels. The B Method, developed by the originator of the Z notation, was specifically focused on using a systematic refinement process to go from specifications to the final code, but this is essentially a manual process, though aided by tools [10]. Using code generation to enforce and/or transform contracts to a lower-level representation is a newer area of research. Approaches based on AADL, SPARK, Simulink[3], and AdaCore's QGen[4] code generator have been part of some of this research [11, 12]. Other research has focused on preserving and analysing other properties, such as code coverage [13] and real-time properties [14]. Some attempts have been made to unify analysis across the modeling and programming levels [15, 16].

The current work is motivated by the integration problems in distributed projects where components are developed by different stakeholders. We are looking for an approach, which allows to pass relevant subset of requirements to each stakeholder in a form that is unambiguous and amendable to formal verification. The main novelty of the approach lies in applying the same set of property specifications in heterogeneous development environment with multiple languages.

2 The Workflow

The approach presented in this paper assumes a top-down workflow in which the initial capture of system requirements is done in SysML[5], followed by algorithm design in Simulink and implementation in C or Ada. However, we find the tools supporting this workflow are also useful in bottom-up or mixed processes.

In the first step of top-down development, the modeler describes the structure of an application (including components, interfaces, and interconnections) in the form of SysML Internal Block Diagrams and requirements in the form of SysML Requirement blocks. At the end of the specification step, the system requirements are fully defined and connected to the components that satisfy each requirement.

[3]https://www.mathworks.com/products/simulink.html
[4]https://www.adacore.com/qgen
[5]https://www.omg.org/spec/SysML

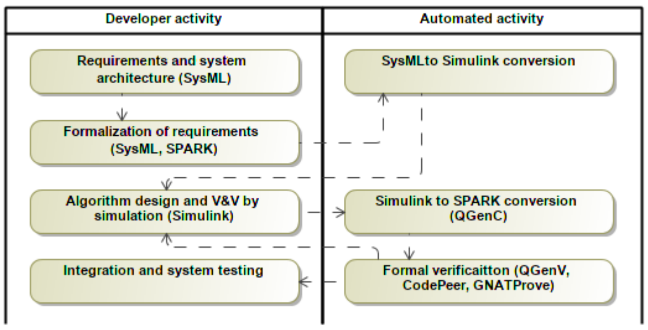

Developer activity
Automated activity
Requirements and system architecture (SysML)
Formalization of requirements (SysML, SPARK)
Algorithm design and V&V by simulation (Simulink)
Integration and system testing
SysMLto Simulink conversion
Simulink to SPARK conversion (QGenC)
Formal verificaitton (QGenV, CodePeer, GNATProve)

Figure 1: Main steps of the workflow

The next step is formalization of the requirements. This activity consists of identifying system-level properties in the textual requirements and rewriting them as constraints. We use SPARK as the constraint language, to simplify the transformation and verification workflow.

After the formalization step, our tooling helps to convert the components and flows from Internal Block Diagrams to Simulink models that are annotated with synchronous observers derived from constraints. This step relies on certain restrictions on the semantics of SysML blocks, which are discussed later in this paper.

SysML to Simulink conversion produces a subsystem hierarchy that reflects the interfaces and connectors defined in the SysML model. The developer then fills this skeleton with algorithms and validates the algorithms by simulation. Finally, when the algorithms are completed and the simulation passes with selected input data, the Simulink to SPARK bridge generates code from the Simulink model and converts observers to SPARK pre- and postconditions. The conversion is similar to the transformation developed in [17], where observers in Simulink are translated first to Lustre and then further to C. In our case we do the translation as one step. GNATprove, SPARK's prover, analyzes this code via the QGen Verifier tool, which traces findings in proofs back to the model. This verification step can be used even when the final implementation language is C. In this case, one uses intermediate SPARK code, which is completely hidden by the QGen Verifier tool, for model verification; once verification is completed, the QGen code generator can be used for obtaining the final implementation in C.

3 The Case-study

This paper builds on a case study designing a demonstrator board with a cruise control simulator for a car. The application consists of a simplified model of car powertrain dynamics (inspired by [18]), a cruise control system, and user interface for manual control of the car.

The case study demonstrates the full application lifecycle from high-level requirements down to implementation. The final application runs on an STM32F4 board and has a physical interface controlling the application.

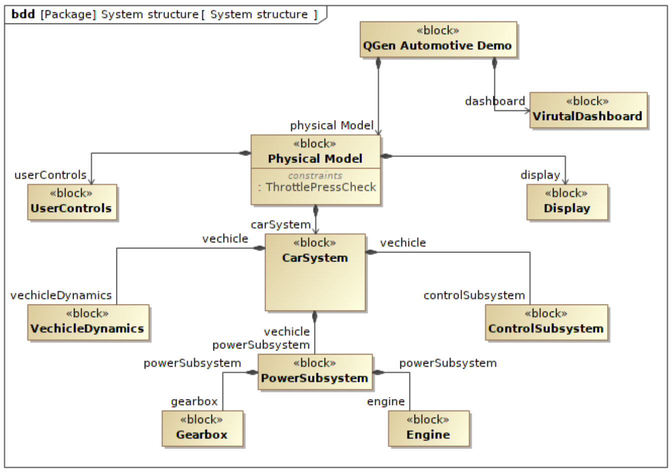


Figure 2: Cruise control simulator components

4 SysML to Simulink

SysML enables systems engineers to collect and organize information about different aspects of a cyber-physical system in one integrated model that describes the relationships between different parts of that system. Later in development, during software design and validation of specified behavior, languages with more dedicated semantics should be used.

The approach taken in this paper relies on Simulink as an algorithm design and simulation tool. The workflow complements the existing transformations with axiomatically defined interface contracts. After defining the system architecture and collecting requirements, the next step is formalization of the requirements. This is done by rewriting the requirements as constraints using the SPARK language and annotating them as Precondition or Postcondition. In other words, we classify the constraints as either assumptions or guarantees.

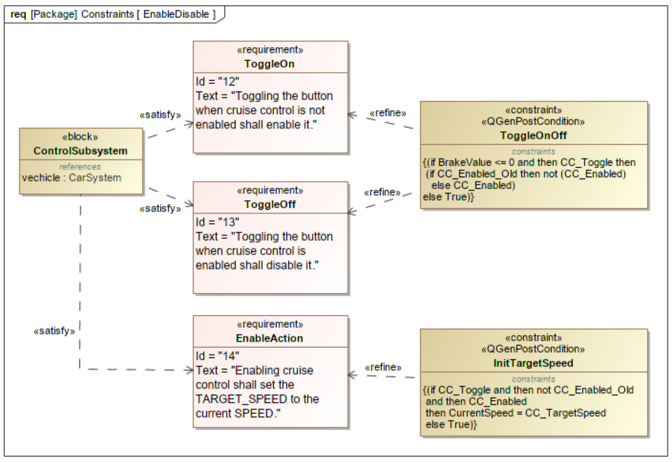

req [Package] Constraints [EnableDisable]
«block» ControlSubsystem
references
vechicle : CarSystem
«satisfy»
«satisfy»
«satisfy»
«requirement» ToggleOn
Id = "12"
Text = "Toggling the button when cruise control is not enabled shall enable it."
«requirement» ToggleOff
Id = "13"
Text = "Toggling the button when cruise control is enabled shall disable it."
«requirement» EnableAction
Id = "14"
Text = "Enabling cruise control shall set the TARGET_SPEED to the current SPEED."
«refine»
«refine»
«refine»
«constraint» «QGenPostCondition» ToggleOnOff
constraints
{(if BrakeValue <= 0 and then CC_Toggle then (if CC_Enabled_Old then not (CC_Enabled) else CC_Enabled) else True)}
«constraint» «QGenPostCondition» InitTargetSpeed
constraints
{(if CC_Toggle and then not CC_Enabled_Old and then CC_Enabled then CurrentSpeed = CC_TargetSpeed else True)}

Figure 3: Refinement of requirements with constraints

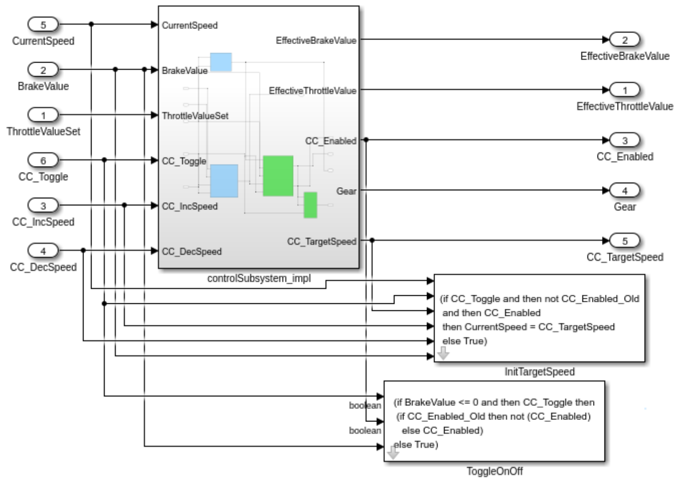
The early prototypes of our SysML to Simulink conversion tool were based on mappings defined specifically for the QGen tool. We are now working on harmonizing the translations with the recent OMG SysPhS[6] standard for linking SysML and simulation environments.

[6]https://www.omg.org/spec/SysPhS

For formalization of the requirements, we considered two languages: the Object Constraint Language (OCL)[7] and SPARK. OCL links naturally with SysML, allowing it to reference elements in properties. While the resulting strong binding between the two languages provides the advantage of clearer specifications, it also requires the models to be more detailed. Our vision is that the first manifestation of requirement-based constraints is rather loosely coupled. The refinement of the constraints happens later, when it is possible to validate them on Simulink models. Given that support for the standard OCL is already built in to case tools supporting SysML, and changing the semantics would be confusing, we decided to rely on SPARK from the beginning of the workflow.

Importantly, in our use of SPARK, we provide a more axiomatic basis to enable modelers to state high-level properties directly at the level of requirements. This approach allows us to talk about behavior even before the behavior has been fully defined. Thus by using a different language, we achieve greater flexibility while simultaneously setting the stage for early, high-level analysis.

Constraints are allocated to components indirectly through requirement blocks, as shown in Figure 3. The transformation from SysML to Simulink converts blocks to Subsystems, Ports to subsystem ports, and constraints to observers. In the case study, constraints attached to the ControlSubsystem are attached to corresponding Simulink subsystem as observers, as shown in Figure 4.



```
(if CC_Toggle and then not CC_Enabled_Old
 and then CC_Enabled
 then CurrentSpeed = CC_TargetSpeed
 else True)
```

```
(if BrakeValue <= 0 and then CC_Toggle then
 (if CC_Enabled_Old then not (CC_Enabled)
  else CC_Enabled)
else True)
```

Figure 4: Control subsystem with observers

5 Simulink to SPARK

The observers composed from the constraints in the SysML model are already executable and can be used for monitoring the simulation once the subsystems expressing functional blocks are complete. However, our final goal is to prove the compliance of the developed algorithms with the design constraints. We do this proof by converting the whole model to SPARK and using GNATprove. During the conversion, the observer subsystems are converted to pre- and postconditions according to their original marking in SysML.

[7]https://www.omg.org/spec/OCL

From the example in Figure 4, the property ToggleOnOff is implemented in Simulink as shown in Figure 5.

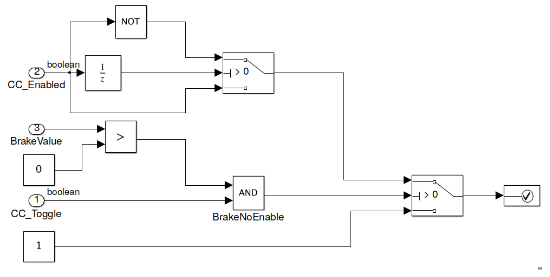


Figure 5: Simulink diagram implementing the ToggleOnOff observer

This property says that when the brake is not active and the toggle button is pressed, the system will toggle its enable state. The check for the brake status is critical, as the cruise control system should not be enabled if the brake is active.

QGen translates the property from Simulink to SPARK like this:

```
function check
  (CC_Toggle : Boolean;
   CC_Enabled : Boolean;
   BrakeValue : Integer_16;
   CC_Enabled_Old : Boolean) return Boolean is
     ( if  BrakeValue <= 0 and then CC_Toggle then
       ( if  CC_Enabled_Old then not (CC_Enabled)
        else CC_Enabled)
    else True);
```

The observer function is attached as a postcondition to the function generated for the ControlSubsystem by the QGen code generator that describes one execution (one step) of the controller. After conversion, we use GNATProve to show that the property holds for the code generated from the model. It is interesting to note that the observer is stateful in Simulink (the UnitDelay block connected to the input “CC_Enabled”) whereas the postcondition is not. Transformation associates the single-step memory to previous computing step and uses ‘Old for obtaining the value.

```
procedure comp
   (ThrottleValueSet :  Integer_16;
    BrakeValue : Integer_16;
    CC_Toggle : Boolean;
    CC_Enabled : out Boolean;
    -- PARAMETERS OF OTHER PORTS NOT SHOWN --
    State  :  in  out controlSubsystem_State)
with Post =>
   (ToggleOnOff.check
     (CC_Toggle, CC_Enabled,
      BrakeValue, CC_Enabled'Old));
```

This approach is somewhat limited, however, and assumes both that the value produced in the output of a computation step is not modified by other processes between two computing steps and and also that past values are not required for input parameters. We are currently working on an extension to this process that will allow access to the full memory of the observed subprogram. This work, as well as the details of the proof, are out of scope for the current paper.

6 Conclusions

This paper demonstrates a workflow for transforming property specifications from a high-level architectural model specified in SysML to Simulink and then from Simulink to SPARK. Once the SPARK code is generated, it is possible to examine the code with static analyzers and proof checkers. The automated workflow from models to code enables the propagation of verification results back to the model, so that they can be used for either correcting the model or refining constraints. Such an integrated approach offers several advantages:

First, we do not expect a full behavioral specification from the systems engineer. The primary goal of the SysML model is to describe the architecture and data flows between different components. Property specifications are attached to architectural components and appear first as loosely coupled constraints. The consistency between different constraints is checked later.

Second, we couple the SysML and Simulink design by transferring the constraints to Simulink as observers and automating the step verifying the compliance of the behavioral specification with the observers.

Third, we use the same generated code both as the means of model verification as well as the means of final implementation. There is no danger that the properties verified on the model are lost in the course of implementation.

Finally, we use of SPARK for property specifications at all levels, from SysML to program code, which ensures consistency of verification results. Although the Simulink uses a set of blocks for executing the observers, the creation of these blocks (and, in future work, also back-propagation of modifications) is fully automated: one can work with the same language throughout the whole process.

There are several opportunities for extending this work. Currently, we are focusing on two topics:

First, we are working on improving the support for reasoning about the past states of the observed model.

Second, we are exploring possibilities to improve the success of automatic proof.

References

[1] J. Abrial, "Data semantics," in *Data Base Management, Proceeding of the IFIP Working Conference Data Base Management, Cargèse, Corsica, France, April 1-5, 1974.*, pp. 1–60, 1974.

[2] L. Lamport, "The temporal logic of actions," *ACM Trans. Program. Lang. Syst.*, vol. 16, pp. 872–923, May 1994.

[3] D. Jackson, "Alloy: A lightweight object modelling notation," *ACM Trans. Softw. Eng. Methodol.*, vol. 11, pp. 256–290, Apr. 2002.

[4] B. Meyer, *Object-oriented software construction*, vol. 2. Prentice hall New York, 1988.

[5] B. Carre, "Program analysis and verification," in *High-integrity software*, pp. 176–197, Springer, 1989.

[6] M. Delahaye, N. Kosmatov, and J. Signoles, "Common specification language for static and dynamic analysis of c programs," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pp. 1230–1235, ACM, 2013.

[7] A. W. Fifarek, L. G. Wagner, J. A. Hoffman, B. D. Rodes, M. A. Aiello, and J. A. Davis, "Spear v2. 0: Formalized past ltl specification and analysis of requirements," in *NASA Formal Methods Symposium*, pp. 420–426, Springer, 2017.

[8] E. T. McGee and J. D. McGregor, "Composition of proof-carrying architectures for cyber-physical systems," in *Proceedings of the 19th International Conference on Software Product Line*, SPLC '15, (New York, NY, USA), pp. 419–426, ACM, 2015.

[9] D. Balasubramanian, G. Pap, H. Nine, G. Karsai, M. Lowry, C. Păsăreanu, and T. Pressburger, "Rapid property specification and checking for model-based formalisms," in *2011 22nd IEEE International Symposium on Rapid System Prototyping*, pp. 121–127, IEEE, 2011.

[10] J.-R. Abrial, "The b-book," 1996.

[11] J. Hugues and C. Garion, "Leveraging ada 2012 and spark 2014 for assessing generated code from aadl models," in *High Integrity Language Technology, HILT 2014*, (Portland, US), pp. 39–46, 2014.

[12] M. Bordin, C. Comar, E. Falis, F. Gasperoni, Y. Moy, E. Richa, and J. Hugues, "System to software integrity: A case study," in *Embedded Real-Time Software and Systems 2014*, (, FR), 2014.

[13] R. Kirner, "Towards preserving model coverage and structural code coverage," *EURASIP Journal on Embedded Systems*, vol. 2009, no. 1, p. 127945, 2009.

[14] I. Dragomir, I. Ober, and C. Percebois, "Contract-based modeling and verification of timed safety requirements within sysml," *Software & Systems Modeling*, vol. 16, pp. 587–624, May 2017.

[15] M. Broy, K. Havelund, and R. Kumar, "Towards a unified view of modeling and programming," in *International Symposium on Leveraging Applications of Formal Methods*, pp. 238–257, Springer, 2016.

[16] V. Bonfiglio, L. Montecchi, F. Rossi, P. Lollini, A. Pataricza, and A. Bondavalli, "Executable models to support automated software fmea," in *2015 IEEE 16th International Symposium on High Assurance Systems Engineering*, pp. 189–196, IEEE, 2015.

[17] A. Dieumegard, P.-L. Garoche, T. Kahsai, A. Taillar, and X. Thirioux, "Compilation of synchronous observers as code contracts," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, (New York, NY, USA), pp. 1933–1939, ACM, 2015.

[18] "Engine timing model with closed loop control, https://se.mathworks.com/help/simulink/slref/engine-timing-model-with-closed-loop-control.html," accessed 2019-04-30.

Scenario-Based Validation & Verification, the ENABLE-S3 Approach

Joan J. Valls, Miguel García-Gordillo, Sergio Sáez
Instituto Tecnológico de Informática, Valencia, Spain; email: {jvalls, miguelgarcia, ssaez}@iti.es

Abstract

Automated systems can be found on many current vehicles, either land, air or maritime. The reliability, safety and robustness of these systems is extremely important, hence validation approaches need to adapt to the ever-evolving necessities of the industry. The ENABLE-S3 architecture addresses the problem of extensive testing by introducing a set of tools and methodologies that can be used to build up a testing environment for different domains. In this manuscript, a special focus is given to the solution developed for the Reconfigurable Video Processor from the aerospace domain.

Keywords: ENABLE-S3, validation, verification, scenario.

1 Introduction

Advances in the development of automated systems can be assessed with the millions of test kilometers that have already been travelled by automated vehicles on public roads. These kind of technologies are leading to improvements in safety, in a more environmental friendly and efficient driving, as well as reducing the number of accidents. Similar statements can also be said for other highly Automated Cyber Physical Systems (ACPS).

Demonstrating the reliability, safety, and robustness of this technology is an arduous task that requires a thorough analysis of the system behaviour under all conceivable situations and potential environmental conditions. There is a lack of cost-effective, commonly accepted verification & validation (V&V) methods. This has been identified as the main roadblock for product homologation, certification and later commercialisation. For instance, some studies [1] state that more than 100 million km of road driving are required to prove that an automated vehicle is statistically as safe as a manually driven one.

In the current digital age, products of the aerospace industry have to fulfill the needs of the user, hence it needs to keep up with the pace of technological developments. Even though aerospace products - specifically satellites and aircrafts - have a very long lifetime, they need to show higher flexibility and better performance in their usage. New solutions are expected to be cheaper, reliable and reach the market in less time. In the case of satellites, for example, this is due to the pressure of NewSpace [2]. The use of Commercial Off-The-Shelf (COTS) components is one of the most evaluated ways to achieve better performance and lower costs. However, their applicability needs to be tested and adapted, e.g. to the space environment, and their suitability in terms of safety and security needs to be checked, e.g. in aerial transport. In the ENABLE-S3 project a lot of effort has been put into addressing the problem in extensive testing these components to guarantee that a sufficient confidence in its reliability and security is achieved. New approaches in system validation are in demand.

The remainder of this manuscript is organised as follows: Section 2 introduces the architecture and methodologies developed during the ENABLE-S3 project. Section 3 presents the difficulties encountered in the aerospace domain and in the Reconfigurable Video Processor use case, particularly. Section 4 describes the scenario-based virtual validation & verification approach followed in the use case and summarises the test framework to evaluate the physical system. Finally, Section 5 offers some concluding remarks.

2 ENABLE-S3 Approach

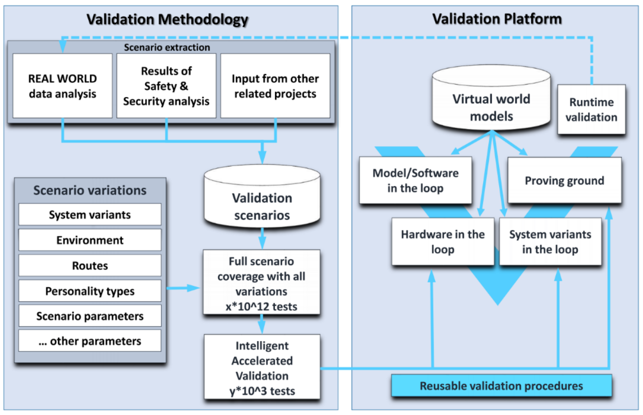


Figure 1: ENABLE-S3 validation toolchain architecture

The aim of the ENABLE-S3 project is to provide the required means for the verification & validation of ACPS. The solution pursued in the project is the identification of relevant scenarios, the automatic derivation of manageable sets of test cases from scenarios as well as the application of automated virtual V&V approaches in combination with physical testing. A consortium of 68 industry and research partners from different application domains (automotive, aerospace, rail, maritime, health and farming) have joined their forces to develop the required technology bricks. Due to this diversity of

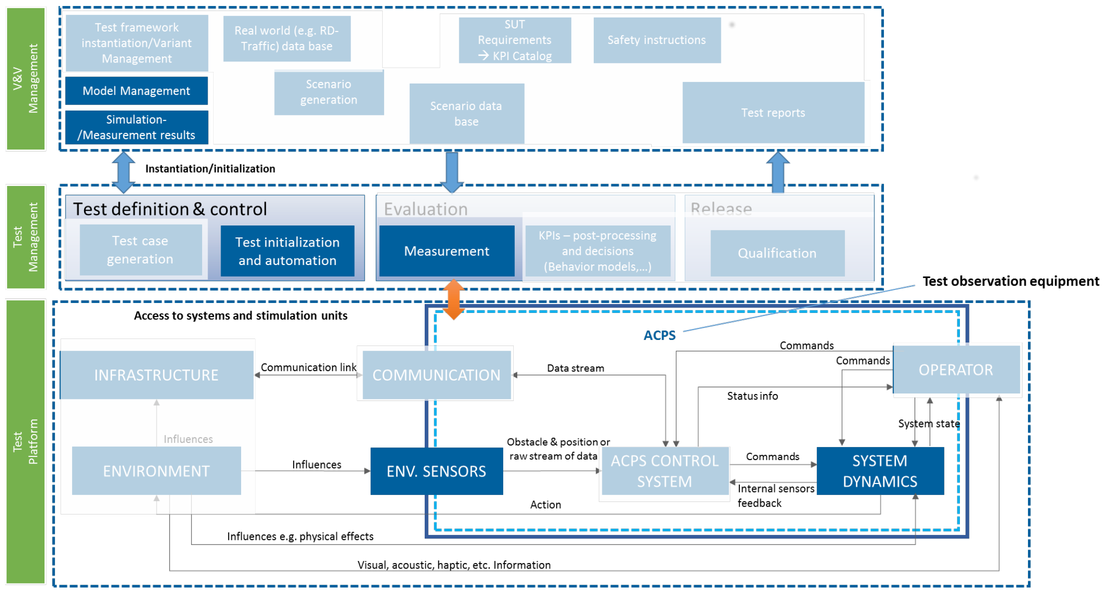

V&V Management
Test framework instantiation/Variant Management
Real world (e.g. RD-Traffic) data base
SUT Requirements → KPI Catalog
Safety instructions
Model Management
Scenario generation
Scenario data base
Test reports
Simulation-/Measurement results
Instantiation/initialization
Test Management
Test definition & control
Test case generation
Test initialization and automation
Evaluation
Measurement
KPIs – post-processing and decisions (Behavior models,...)
Release
Qualification
Test observation equipment
Test Platform
Access to systems and stimulation units
ACPS
Commands
Commands
INFRASTRUCTURE
Communication link
COMMUNICATION
Data stream
OPERATOR
Status info
System state
Influences
ENVIRONMENT
Influences
ENV. SENSORS
Obstacle & position or raw stream of data
ACPS CONTROL SYSTEM
Commands
SYSTEM DYNAMICS
Internal sensors feedback
Action
Influences e.g. physical effects
Visual, acoustic, haptic, etc. Information

Figure 2: ENABLE-S3 testing environment in the aerospace use case

partners and application domains, the project does not aim for a single common, generic software solution. The idea pursued was the development of a common methodology, a basic verification and validation toolchain architecture (Figure 1) and a set of reusable technology bricks, which can be used to build up a testing environment for use cases in different industry domains (Figure 2). The complete results of the project have been published [3]. The remainder of this manuscript summarises the work done in the aerospace domain use case. More detailed information on this use case can be found in [4].

3 Aerospace Domain Use Case

The main focus of the aerospace domain use case is the improvements in the validation and verification processes of a Reconfigurable Video Processor (RVP) that will be sent to space missions.

One of the main problems of autonomy in space applications is that once a mission is in orbit it is very difficult, even neigh impossible, to replace a processing module on board. Application-Specific Integrated Circuits (ASICs) and antifuse-based FPGAs are the most common solutions for the vast majority of space digital systems. Problems related to these technologies are the non-recurring engineering costs and the lack of flexibility that is demanded in the NewSpace which is currently being defined. For that reason, SRAM-based FPGAs that serve as programmable devices with shorter design cycles and reduced NRE could alleviate the aforementioned inconveniences. Additionally, modern FPGAs also offer the possibility to be reprogrammed on-the-fly which makes them more interesting for remote long-term space missions.

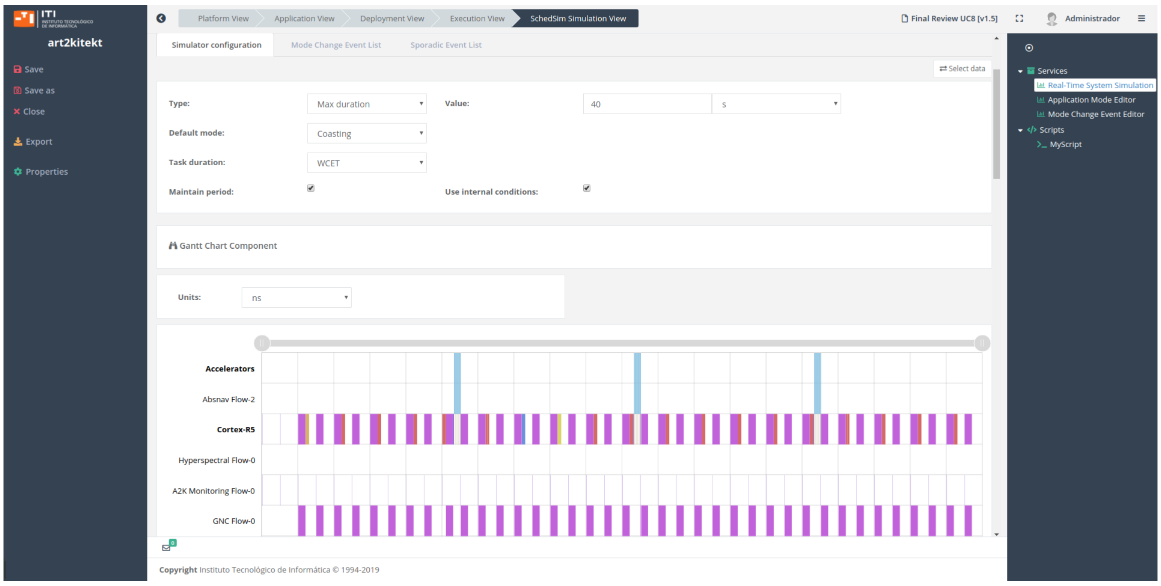
However, in space missions, there are some unique environmental challenges that need to be accounted for and that may have a large impact on the utilization of these technologies. Despite the high performance, flexibility and low design costs, the volatile nature of SRAM-based FPGAs makes them highly susceptible to radiation effects. When a particle hits and SRAM-based device, the content of one or several cells may change. When this event happens, the implemented functionality may also change which can have catastrophic consequences. Hence, commercial parts employing these FPGAs cannot provide a reliable hardware for space environment since they have not been designed following secure radiation-hardened and fault-tolerant design processes.

The ARTICo3 architecture [5] is a hardware-based processing architecture for high-performance embedded reconfigurable computing. It uses Dynamic and Partial Reconfiguration (DPR) in SRAM-based FPGAs as its technological foundation. The architecture supports and enables run-time adaptable implementations of data-parallel algorithms. Two different types of parallelism can be exploited using ARTICo3: (1) by using several replicas of the same hardware accelerator it provides data-level parallelism, and (2) by using different hardware accelerators it enables task-level parallelism. Although these features offer fault tolerance in the reconfigurable partition, additional mechanisms are required in a safety-critical context, as the space scenario of the use case.

To correct transient faults in memories, some techniques, generally called scrubbers, are utilised. In the RVP they have been implemented in different layers: real-time processors (ARM Cortex R5), platform management unit (PMU), and dedicated hardware cores inside the FPGA.

A wide range of Cyber-Physical Systems (CPS) applications can be developed with the ARTICo3 framework. The applications should support HW/SW partitioning, so that certain tasks can be offloaded to and executed in hardware accelerators. Two algorithms have been implemented in regards to the use case, which follow the requirements of the framework.

On-board image compression techniques are mandatory in space remote sensing missions to reduce data size prior to



Type: Max duration
Value: 40 s
Default mode: Coasting
Task duration: WCET
Maintain period: ✓
Use internal conditions: ✓
Units: ns

Figure 3: Art2kitekt Scheduling Simulation Tool

sending them to ground stations where they are processed. In this case, a lossy extension of the Consultative Committee for Space Data Systems 123.0-B-1 Lossless Multispectral and Hyperspectral Image Compression algorithm has been developed [6]. The algorithm provides a trade-off between its compression efficiency and the design complexity. The output data constitute a variable-length encoded bitstream from which the original image can be fully recovered. It uses a scheme based on prediction and entropy coding of the resultant prediction residuals, i.e. the differences between each input sample and its corresponding prediction value.

Guidance, Navigation and Control engines will allow autonomous navigation of spacecrafts, including traveling to planets or asteroids surfaces, orbiting around those stellar bodies, etc. Accurate positioning is required to enable pinpoint landing ability. Several vision-based navigation algorithms have been implemented, and since they are computationally intensive, the hardware acceleration that can be obtained thanks to the accelerators in ARTICo3 will be essential. Specifically the algorithms that have been developed are: Absolute Navigation Algorithm, Relative Navigation Algorithm, and Stereo Vision Algorithm.

4 Scenario-based virtual V&V

Since the RVP for space missions cannot be validated under real conditions, a scenario-based virtual validation campaign is performed. This virtual V&V campaign aligns to the following main objectives of ENABLE-S3:

- reduction of expensive testing time in nuclear facilities;
- optimization of the test setup through lab testing and prior to real radiation testing;
- strengthen critical parts of the design in early stages of the development;
- optimization of the component qualification effort.

4.1 Model-in-the-Loop

First step of the validation will be done at the model level. The RVP is modelled and analysed in order to discard unfeasible scenarios early on. A test scenario is planned, inspired by the different stages a spacecraft may encounter during a real space mission. The model includes the adaption capabilities of the RVP, i.e. the recovery mechanism/fault mitigation systems and the reconfiguration mechanisms to handle the different operational modes and its transitions.

4.1.1 Platform and application models

The art2kitekt tool suite is used to assist in the V&V process, more concretely, its modelling features that help in the design process of high integrity systems with real-time constraints.

A collection of components such as processors, memories, buses and devices are offered by the framework and allows the engineer to build an heterogeneous system that suits their needs. The desired system model is built by creating instances of defined component types (e.g. the user can define their own processor types) and by connecting them through buses. The current version of the framework allows the instantiation of processing devices, i.e. components that are not processors but which are able to execute code (e.g. an accelerator). This also includes the programmable devices part of the ARTICo3 architecture.

The specification model that describes the application consists of a series of independent execution flows comprised of several activities. In some contexts they are referred to as end-to-end flows and tasks. These flows have different activation patterns, i.e. either a periodic activation or an sporadic activation after certain event has been issued. Each flow has several activities that model the inner functionality and the precedence relationships between them. All activities can be refined from coarse to fine-grain modelling as the engineer considers necessary.

4.1.2 Scheduling Simulation Tool

Schedulability analysis is one of the most important evaluations of a system, especially in hard real-time systems [7, 8], in which missing a deadline can lead to catastrophic consequences. There are several formal techniques to analyse the worst-case response time of tasks [9]. In some scenarios, these techniques do not always provide exact solutions, for example in distributed hard real-time systems [10]. They work with assumptions such as all tasks being independent or requiring some restrictions in order to apply them, hence results are pessimistic. This, in turn, brings the inefficient use of the computing power of real-time systems in order to guarantee the feasibility of its schedulability.

A complementary approach to performing off-line analysis with formal techniques is the evaluation of the real-time system through simulation of its real-time behaviour. Although simulation cannot assure the validation of a system, it can help with the study and understanding of its behaviour and limits. Some of the advantages offered by simulation that make for an interesting tool are:

- there is no probe effect (no disturbance) on the system due to instrumentation, which can be problematic in real-time systems;
- the engineer can replay scenarios with ease in order to evaluate how changing the studied tasks, or even the executing platform, may affect the results;
- the effect of non-static or non predictable events can be studied.

A new scheduling simulation tool has been integrated into the art2kitekt framework, a screenshot of which is depicted in Figure 3. The implemented tool adapts to the specific requirement of the use case. For instance, it has the capability of simulating non periodic events such as the presence of a fault in the system due to radiation. Also, the tool incorporates functional modes and its mode changes, in order to represent and analyse the different behaviour of the system during each of its mission phases, since it is essential to guarantee the fulfilment of the deadlines not only during each operational mode, but also during the transitions between them. In this manner, the engineer can model several scenarios to have a more detailed analysis and this leads to a reduction of the test procedure by detecting unfeasible scenarios in early stages of the development.

4.2 Hardware-in-the-Loop

In the next steps, the aforementioned simulation models must be implemented using functional software for testing in the final hardware. The model components are step by step replaced by software components (SiL, Software-in-the-Loop) and executed in the real hardware components (HiL, Hardware-in-the-Loop), and finally, the overall system is tested. This methodology requires a flexible and safety simulation framework with the capability of communicating with the hardware platform, complying with the real-time requirements of the System Under Test (SUT). This framework should help in integrating the different technology bricks involved in the tests and in collecting the results provided by them.

4.2.1 Test Framework

The ENABLE-S3 architecture defines the generic test framework that is comprised of the test execution platform and the test management. Different applications are involved in these processes and are coordinated by art2kitekt, in order to generate the test scenarios and to execute the SUT with the required test configurations.

Figure 4 depicts the validation platform in the aerospace use case that has been developed over a board based on a Xilinx Zynq Ultrascale+ MPSoC. It is composed of (1) a Real-Time Processing Unit (RPU), (2) a Programmable Logic FPGA-based unit (PL), and (3) a general Application Processing Unit (APU). The RPU and the PL are in charge of executing the SUT, and the APU is responsible of executing the embedded part of the Test System (TS). Both, the SUT and the TS, coexist in the same system but running in different domains.

A bidirectional communication protocol between the TS and the SUT has been implemented thanks to the capabilities of the MPSoC: the shared memory and the Inter Processor Interrupt device. With the aim of sending large messages between processors using a zero-copy approach and a low-latency transmission. This communication permits (1) to configure the SUT, (2) to coordinate the test execution with the Test System, and (3) to collect the output measures to allow later processing and visualization.

The test scenario is composed of several Application Processes (APs) running in the TS processor, that are connected with their counterpart in the SUT processor. The different APs involved in the tests are the gateway between the SUT and the external applications, in charge of simulating the sensors and the data providers necessary to generate the scenarios. Thanks to this architecture, the TS provides an interface to help in the integration of independent test applications.

Furthermore, the system coordinates the different APs and the SUT, using a common command interface, that permits to configure the whole system with the same parameters and to coordinate its execution. The TS coordinator, based on a state machine approach, allows the application to be commanded, waiting for them and avoiding APs from breaking the test timing.

In the aforementioned test platform, art2kitekt offers both, the low level architecture to build the test system, and the user interface to configure the scenarios and to visualise the test results. This tool suite has the goal of helping the engineer to control the system in the whole V&V process.

4.2.2 Monitoring Tool

Observing the behaviour of a real-time system helps the engineer in several ways in the V&V process. In early phases of the development, observed execution time, mixed with the estimated one, can be used to analyse the system and to ensure the fulfilment of the temporal constraints. In addition, comparing the observation with the expected behaviour helps to find incorrect implementations and to improve the initial model.

The art2kitekt toolsuite has been extended to provide a run-time monitoring tool [11] with the capability of collecting

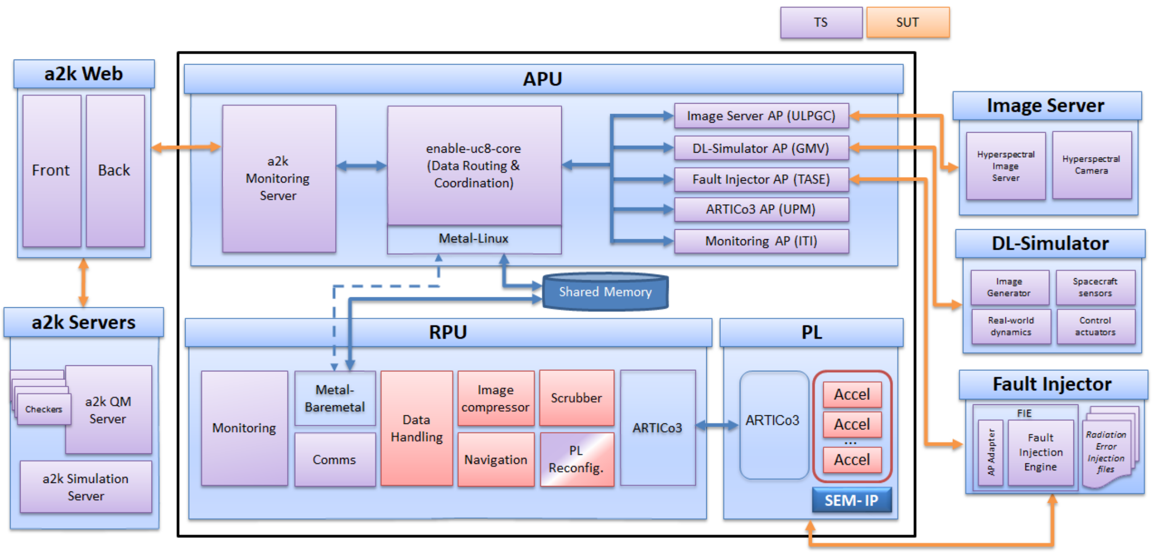


Figure 4: Test framework architecture

statistics of the SUT, of processing them, and of displaying the results in a unified interface.

This new service, integrated into the Test Framework (see section 4.2.1), has been made with the purpose (1) of finding system events not taken into account in the initial model, (2) of obtaining a better approximation of the system performance, and (3) of computing the observed temporal behaviour, defined by the Observed Worst-Case Execution Time and the Observed Worst-Case Response Time.

The monitoring tool allows the engineer to discard unfeasible temporal configurations of the SUT and to define the component implementations to be optimised. It also allows collecting application specific data, such as fault conditions or the use of shared resources, with the aim of validating the system executions.

An intrusive but small overhead software technique has been developed to collect the necessary data and to compute the temporal information of the test executions. A trace recorder component must be added to the SUT in order to be able to store the different temporal events and the time when they have been executed. The implementation complies with the Test Framework interfaces and with its internal communication protocol.

4.2.3 External applications

In order to build the overall test scenario, external applications are connected to the test framework using the APs interface. They must provide the necessary information to simulate the environment or to analyse the output data, that can be used in a close-loop simulation or to verify the system behaviour.

Fault Injector The possibility of injecting real faults in the FPGA, according to the expected failure obtained by analyzing the radiation environment, allows the evaluation of the behaviour of the designs installed in the SUT in the different stages of development. The fault injector permits the development of test set-ups adapted to the failures expected. This, in turn, reduces the number of visits to the BEAM, the testing time, and, in general, provides robustness against radiation to the design.

The Fault Injection Engine is a mix of the hardware on the actual device in which the engineer wants to inject faults and a software, which is capable of controlling this hardware. Verification and validation are essential steps in the development process of any autonomous system and as such represent key targets of ENABLE-S3.

Image server The on-board image compression algorithm, implemented in the exposed use case, must be connected to a real camera, provider of the hyperspectral images to be compressed. In order to emulate this behaviour, an hyperspectral image server is connected to the test framework via AP, working as an input sensor from the point of view of the SUT.

DL-Simulator The vision-based navigation algorithms integrated in the SUT needs, not only the hyperspectral images provided by the image server, but also the spacecraft sensors data, in order to execute the navigation filter and to manage the control actuators. For this reason, a specific space simulator (DL-Simulator) has been integrated in close-loop into the test system using the APs, providing the data of the emulated sensors to the SUT and using the actuator signals generated by the on-board navigation to manage the simulated spacecraft.

5 Concluding Remarks

The developments achieved in the aerospace use cases in ENABLE-S3 allow testing the applicability and suitability of COTS extensively. Moreover, evaluating their performance, by simulating the physical environment and assessing the safe and secure placing before testing under real conditions, reduces the cost of test campaigns and to acquire a greater knowledge about the behaviour of the system. This opens the possibility of new more flexible developments, with high performance and at a lower cost than using the traditional approach of aerospace.

Acknowledgements

This work has been conducted within the ENABLE-S3 project that has received funding from the ECSEL Joint Undertaking under grant agreement No 692455. This joint undertaking receives support from the European Unions HORIZON 2020 research and innovation programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

References

[1] H. Winner and W. Wachenfeld, "Absicherung automatischen Fahrens, 6," *FAS-Tagung München, Munich*, vol. 9, 2013.

[2] G. Martin, "NewSpace: The emerging commercial space industry," tech. rep., NASA Ames Research Center, 2015.

[3] A. Leitner, D. Watzenig, and J. Ibanez-Guzman, *Validation and Verification of Automated Systems: Results of the ENABLE-S3 Project*. Springer International Publishing, 2019.

[4] L. Armesto Caride, A. Rodríguez, A. Pérez Garcia, S. Sáez, J. Valls, Y. Barrios, A. J. Sanchez Clemente, D. González Arjona, Á. J.-P. Herrera, and F. Veljković, *Reconfigurable Video Processor for Space*, pp. 231–249. Validation and Verification of Automated Systems: Results of the ENABLE-S3 Project, Springer International Publishing, 2019.

[5] A. Rodríguez, J. Valverde, J. Portilla, A. Otero, T. Riesgo, and E. de la Torre, "FPGA-Based High-Performance Embedded Systems for Adaptive Edge Computing in Cyber-Physical Systems: The ARTICo3 Framework," *Sensors*, vol. 18, no. 6, p. 1877, 2018.

[6] R. Guerra, M. Díaz, Y. Barrios, S. López, and R. Sarmiento, "A Hardware-Friendly Algorithm for the On-Board Compression of Hyperspectral Images," in *2018 9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1–5, IEEE, 2018.

[7] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-time systems*, vol. 28, no. 2-3, pp. 101–155, 2004.

[8] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, vol. 24. Springer Science & Business Media, 2011.

[9] M. H. Klein, T. Ralya, B. Pollak, R. Obenza, and M. G. Harbour, *A Practitioner's Handbook for Real-time Analysis*. Norwell, MA, USA: Kluwer Academic Publishers, 1993.

[10] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocess. Microprogram.*, vol. 40, pp. 117–134, Apr. 1994.

[11] M. García-Gordillo, J. J. Valls, and S. Sáez, "Heterogeneous Runtime Monitoring for Real-Time Systems with art2kitekt," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 266–273, Sep. 2019.

Maintaining Trust in VANETs Using Blockchain

Ravi Tomar

School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India; email: ravitomar7@gmail.com

Sarishma

M. Tech. Computer Science and Engineering Faculty of Technology, Uttarakhand Technical University, Dehradun, India; email: sarishmasingh@gmail.com

Abstract

Vehicular ad-hoc networks are networks formed by fast moving vehicles which come in contact momentarily and exchange information. Since it's an ad-hoc network, it becomes difficult to maintain trust, security and authenticity of information being exchanged in the network. In this paper, we leverage the concepts of blockchain to maintain trust in the network. Since blockchain provides a tamperproof, decentralized mechanism to store data, we use it to store information related to events such as collision, accident, SOS etc. The information stored on blockchain can be used to validate it at later points of time so as to minimize the false benefit cases by use of Proof of Location certificates. The proposed system has the potential to increase the trust of end users in VANETs. It can also be integrated into the design of future vehicles because of its ease of implementation. The paper also discusses the benefits and constraints of the proposed model along with the related future work.

Keywords: Vehicular Ad-hoc Networks, blockchain, Proof of Location.

1 Introduction

Information and Communication Technology have penetrated all the major spheres of human life in the recent times. Vehicular networks form one such case where vehicles are being equipped with smart devices so as to enable exchange of information between vehicles. Exchanging right information at right time may lead to numerous benefits such as saving human lives, reducing traffic overhead, decrease in resource consumption etc. In a vehicular ad-hoc network, vehicles act as collectors as well as disseminators of information and connect to one another in an ad-hoc manner [1]. The vehicles are fast moving, highly dynamic entities which restrict the time of contact between vehicles. This time restriction leads to compromise with respect to security in the network. The information is passed between vehicles but there is hardly enough time to authenticate and validate the identification of the sender and the information sent. This constitutes as one of the serious drawbacks when it comes to VANETs.

Concept of blockchain was first introduced in 2008 by Satoshi Nakamoto [2] in a white paper. Around 2012, its applications revolutionized the digital transaction environment and soon transformed itself into one of the securest mechanisms to do business. Blockchain can be thought of as a data structure which stores data in a timestamped manner making it tamperproof as data once written cannot be changed at later points of time. Today blockchain finds application in fields such as health, finance, asset management, logistics, industries, Internet of things, ownership rights [3], etc.

Vehicular ad-hoc networks are commanding a great deal of research in the current scenario as they will give way to what we call as Intelligent Transport Systems. However, securing the network and validating the information still remains a challenge. Considering the many benefits of using blockchain, in this paper we propose to implement it with respect to Vehicular ad-hoc networks. The aim is to make the networks more secure, information more reliable and to identify malicious nodes or outliers. We propose a detailed system model which will work on an algorithm to tailor the blockchain concept with respect to VANET's needs. We also propose potential applications of the concept which can help in significant reduction of monetary losses occurring due to false cases and behavior of users.

The rest of the paper is organized as follows: Section II outlines the related work where we focus upon the work already which has been done on the topic. Section III and IV covers the background of Vehicular ad-hoc networks and Blockchain respectively. Section V gives our proposed work along with assumptions, system model and the associated algorithm. Section VI covers the main applications where the concept can be utilized. The benefits of the proposed work and the challenges are presented in Section VII. Future work which can be done in the area is then given in Section VIII, after which the paper is concluded.

2 Related Work

In this section, we explore the previous work done in the field of integration of VANETs and Blockchain. VANETs as well as Blockchain have attracted tremendous research focus on an individual basis but not much has been done in the grey area where both can be integrated. Some of the work which has been done include:

- R. Shrestha et al. [4] propose a real world public blockchain which focuses on trustworthiness of nodes by adjusting the factor everytime nodes misbehave or generate false information.
- B. Leiding et al. [5] proposes Traffic Regulation Application where traffic regulation can take place using blockchain via Ethereum's programming languages. The vehicle can drive autonomously by basing itself entirely on the information available on blockchain. This will also enable punishing of misbehaving vehicles and imbibe more regulated driving. They also propose paying of vehicle tax via blockchain which can maintain payment records more efficiently. Many countries require payment of annual vehicle taxes or other bills. Ethereum linked accounts can be used from which automatic deduction can take place.
- W. Liu et al. [6] propose a blockchain based system named as BARS which provides an established trust model for VANETs. It also gives a reputation evaluation algorithm which tells about the reputation of the vehicles.
- P. K. Sharma et al. [7] propose SmartPay which uses the blockchain to assist the vehicle's driver by assisting him/her in running errands like refuelling, shopping groceries etc. The payments and routes are decided beforehand via carefully drafted smart contracts.
- M. Singh et al. [8] present a reward based system where vehicles are rewarded with credit points which can be later used to pay financial bills online.
- SmartShare is a real time ride sharing application [7] which uses the Block-VN model to provide shared rides. The idea is to utilize the private transport frameworks so as to leverage wasted space, seats and payload space.
- Rakesh Shrestha et al. [9] propose the creation of a local blockchain which can store node trustworthiness and message trustworthiness of the network.

3 Blockchain

The concept of Blockchain was introduced by Satoshi Nakamoto [2] in a white paper in 2008. It was implemented in cryptocurrency and economic transaction domain as it solves the double spending problem through which it gained immense popularity [3]. Currently, use of blockchain has expanded to many dimensions such as healthcare, asset management, financial and non financial applications, smart contracts, government policies [10] etc.

Blockchain refers to the chain of blocks which works as a distributed ledger in a decentralized environment. It is a form of data structure where data once stored cannot be erased or tempered with. Block serves as a fundamental unit which stores records or transactions in a timestamped manner. Each block has only one parent block. The first block of the blockchain is called as Genesis block [11] which has no parent. The transactions (or events or records) are accumulated over a time window and are then mined as a block. Mined block contains hash value of its previous block thereby making it a connected chain. Each block also contains a Merkle tree root hash [12] which is a hash value of all the cumulated transactions. This makes it even more difficult to alter data contained in blockchain as any change will lead to change in the overall hash value of the blocks which can be easily detected.

Block contains [3] a block header and a block body. Header comprises of block version, merkle tree root hash, timestamp, nonce and parent block hash value. Body of the block contains all the transactions in a chronological manner along with a transaction counter. Blockchain uses asymmetric cryptography to authenticate transactions. Each user is issued a pair of private and public keys where former is used to sign a transaction and latter is used to verify the transaction.

Another key concept in working of blockchain is of attaining consensus among nodes as to which block should be mined. All the nodes or most of them should agree on one block before mining. Blockchain offers many approaches to reach consensus such as Proof of Work [2], Proof of Stake [13], Practical Byzantine Fault Tolerance [14], Delegated Proof of Stake [13], Ripple [15], Tendermint [16], etc.

The most popular among these is Proof of Work where a node which wants to mine a block has to solve a computation intensive problem first. When a node spends considerable amount of time in solving the problem, then we assume that the node is safe and is not likely to attack the blockchain. Thus by showing Proof of Work, the node is allowed to mine the block to the blockchain. Key characteristics of blockchain include decentralization, auditability, transparency, security, integrity, security, tamper proof etc.

4 VANETs

VANET stands for vehicular ad-hoc networks which are derived from the concept of Mobile Ad-hoc Networks. VANETs are more dynamic in nature than MANETs. Stationary or fast moving vehicles come together in an ad-hoc manner to form VANETs. Characteristics of VANETs such as dynamic topology, very less time of contact, routing of information etc. make it somewhat difficult to manage. VANETs have come up as an emerging base for the development of Intelligent Transport Systems [17].

There are three main components associated with vehicles in a VANET. First is On Board Unit or OBU which is typically mounted on the vehicle for information exchange between other OBUs and Road Side Units (RSUs). Communication is done via short range wireless communications such as Dedicated Short Range Communication (DSRC) or WAVE (Wireless Access in Vehicular Environments). Second component is Autonomous Unit or AU which utilizes the information accumulated by the OBU. It might be mounted along with OBU or can be a separate mobile device such as handheld devices or laptops. Third component is Road Side Units or

RSUs which are stationary infrastructure units used to store, forward, exchange information and to assist vehicles in other activities.

VANETs have gained popularity because they have the potential to make use of the vast amount of data which they sense while they move. By exchanging this data, they can make the user experience much better and can lead the way to autonomous learning behaviour of vehicles. However data dissemination has been a big challenge for VANETs, primarily because of a very short duration of contact among vehicles, during which they have to exchange data. This time restriction takes a heavy toll on security of the network and the data.

5 Proposed blockchain based VANET

We propose a system whereby events such as road accident, hazards, traffic jams, rash driving etc. are recorded by vehicles and the exact information is stored on the blockchain in the form of time-stamped blocks. We propose maintaining two blockchain's, one on the vehicle itself which will record every activity and the other on the geographical level which will be maintained at the cloud end. The blockchain will be geographically maintained either at a country level or at region level.

The country area can be divided into major regions depending upon the number of vehicles running on road. As information pertaining to one region might not be of much use to other regions, we can maintain regional blockchain's efficiently [9]. There definitely is a trade-off when it comes to maintaining two blockchain's at different ends. Optimization of these blockchain's is left as a future work to be done in this area.

For vehicles migrating from one region to other, a small buffer area can be created to check for the information. Floating genesis block can also help in getting updated blockchain when vehicles move from one region to another. The information is validated at various stages before mining the block. Since data once written on a blockchain cannot be rolled back, we can use it at later points of time for data validation and checking.

In order to implement the proposed system, there are some areas which need clarification. We give the details with respect to following features:

- Participating Network: A fast changing network made of vehicles where vehicles will act as nodes. Each node will act either as an endorser or a committer.
- Integration with existing system: The functionality of BCU can be performed by OBU as it has the required resources to assume the role. However, the security might be compromised if this happens. Installation of a new unit – Blockchain Unit with strict data entry and exit rules will provide a more secure mechanism for the operation of the system. BCU will be designed such that tampering with its code is extremely difficult.
- Actors and their role: The main actors in the system will be vehicles (endorser and committer), Road Side Units, Regulating Authority, Cloud end storage and back-end analysis modules. The vehicles will act in one of the following roles at a time: Endorsers are the vehicles which are an active party to the event. They will be the ones who will initiate the transaction. The transaction will contain the parameters related to the current state of the vehicles. RSU will record the endorsed transactions. Committers are the vehicles who will receive the transaction broadcasted by the endorsers. They will then commit to the transaction and sign it using their own key. The transaction is then again broadcasted. RSU will record the committed transactions and issue a location certificate in the form of Proof of Location (PoL) to all the committers who sent a signed transaction.
- Rules associated: Any vehicle can assume only one role at a time. Block mining will depend on the density of the region around the RSU. RSU will act as an ordering service and will be responsible for chronological ordering of transactions.
- Validation of data: The data will be validated at three stages. Firstly by the committers who will commit only when the data is correct. Second by the RSU who will analyze all the transaction before making the decision of block mining. Thirdly, the data can be cross checked by the analysis modules present at the cloud storage.
- Data added/appended at which stages: Data will be added to the endorser blockchain unit when a transaction is endorsed. Committers will add the data in both the cases, if they commit or if they don't commit. This way the data about the vehicle reporting false event information will be recorded. The data will be stored at the regional blockchain in a permanent manner in the form of blocks.
- Permission access: Trust Level is the factor which identifies which vehicles are trustworthy and which ones are malicious. After crossing a threshold value of Trust Level, the vehicle will have to drop out of the endorsing role and contact the regulating authority for further action.

5.1 Assumptions

There are some assumptions which are listed below followed by a detailed system model and the associated algorithm for the same.

We assume that all the vehicles can communicate with the in-range vehicles and are connected to the internet. Vehicles have a unique ID issued by a trusted authority along with a pair of public and private keys. The messages exchanged are time-stamped and digitally signed. We assume that each vehicle is equipped with Geographical Positioning System (GPS), On-Board Unit and a custom made Blockchain Unit (BCU). We also assume that the Road Side Units are not malicious in nature, or to the very least two out of three are not malicious in nature. We assume there is a central storage repository on cloud which is in synchronization with Vehicle Regulation authority.

5.2 System model

All the vehicles will be associated with a Vehicle ID, pair of public-private keys and a parameter - trust level. Vehicles will constantly sense and store the environment variables in their OBUs in a temporary manner. Parameters from this information are sent as beacons to neighboring vehicles. BCU will remain in sync with the OBU and will store the records permanently. Sensing and storing will continue as long as no event occurs. When an event occurs such as a traffic jam, a collision or a hazard, the information sensed will be sent by the vehicles to the nearest Road Side Unit(s).

The information here will work as a transaction for the blockchain. The variables sensed will be sent to BCU by the OBU which will be responsible to generate the transaction. The transaction will contain the digital signature, vehicle ID, sensed parameters, timestamp and a hash of sensed values to avoid tampering by other vehicles. The transaction will of the following form:

- TrVID = (info, x, y, z, s1, s2, dir, r, ts, ds, hID) Where, x, y, z are the location coordinates.
- ‘s1, s2 ‘ and dir represent initial speed, current speed and direction of motion, respectively.
- ‘r’ represents the region in which the vehicle is present. Value of r can be varied for various locations, depending upon the classification of regions.
- ‘ts’ and ‘ds’ represent timestamp and digital signaturr, respectively.
- ‘hID’ is the hash value of the entire transactions being sent.

Vehicles assuming the role of endorsers will be the ones who will broadcast the transaction. RSU will record it and issue location certificates as PoL to them. Committers will also receive the same transaction; they will check the data and commit to the transaction by signing it. Then again, they will broadcast the transaction to RSU. This data will be crosschecked by the RSU. First step is to check whether the location of the vehicle is correct or not. After receiving the information, RSU will issue the PoL certificate to the vehicle. This will work as a verification mechanism for the vehicle which will then append this certificate to its own personal Blockchain residing on the BCU. Location certificate issued by RSU acts as a digital proof for the vehicle which certifies that the vehicle was present at the geographical position when the event occurred.

RSU will wait for ‘n’ time so that it receives information update from all the vehicles that are nearby. The maximum value of n will be 30 minutes. However depending upon the density of vehicles in the area, this value of time will reduce significantly. Once the time window closes, RSU will analyze all the transactions. If the data sent by the vehicles matches with one another, then RSU assumes that event has indeed happened. If the information received does not match and is contradicting then the information is mined depicting it was falsely stated by the vehicles along with their Vehicle IDs. If the data is found to be correct, it mines the information on to the cloud based regional Blockchain. If the data is found to be inconsistent, then it is discarded and the trust level of associated vehicles is decremented. Trust level of each vehicle is calculated as:

$$TL= m/(m+n)$$

If the reported event is correct then m is incremented by 1 else n is incremented by 1 [4].

The blockchain can be updated whenever the vehicles join the network or when they change the operating regions. A floating genesis block will be made available at all the RSUs through which vehicles can access the current blockchain.

5.3 Algorithm

- RSU constantly listens to beacons and packets exchanged.
- Whenever a Vehicle is turned ON, BCU unit is turned ON; beacons are sent to all the neighbours
- RSU senses the number of vehicles per unit area according to the beacons received and calculates n as

  $$n = h \text{ (area / number of vehicles)}$$

  where h is a constant of proportionality
- When any event happens,
  - ➢ Nodes in direct participation assume the role of endorsers.
  - ➢ Endorsers sense the data via OBU.
  - ➢ OBU passes this data to BCU.
  - ➢ BCU generates a new transaction as Tr_{VID} = (info, x, y, z, s_1, s_2, dir, r, ts, ds, h_{ID}).
  - ➢ BCU sends it to OBU and OBU broadcasts it to its neighbourhood.
  - ➢ Other nodes either commit to the transaction or they don’t
    - o Committers compare the information by passing it to their own BCUs.
    - o Committers assigns Y or N value for correct or incorrect information respectively and then sign it using their key($Sign_{VID}$).
    - o After signing, another hash is calculated to avoid tampering with the information.
    - o Tr_{VID} = (info, x, y, z, s_1, s_2, dir, r, ts, ds, h_{ID} , (Y/N) $Sign_{VID}$, h_{ID}).
    - o RSU on receiving it, sends a PoL to the associated VID.
    - o Committer sends this PoL to the BCU which appends its to its own blockchain.
- After ‘n’ time window is over, RSU pushes the data onto cloud whereby further analysis takes place.
- Trust Level of vehicles is generated and passed to RSU by the cloud server backend applications.
- RSU updates its blockchain and beacons other vehicles to update their blockchain as well.
- RSU senses the environment again, calculates the value of ‘n’ and the cycle repeats.

6 Applications of the proposed work

The concept of blockchain integrated to VANETs provides us with more secure and reliable VANETs. We thus present some applications where this concept can be used to yield maximum benefit.

1. Insurance companies: Insurance companies issue credit to those users who satisfy their terms and conditions. Many times, users use fake details in order to avail the benefits from the companies. This leads to monetary losses on behalf of companies. Also at times, genuine users suffer because too much inspection is done before offering credit. By using blockchain, the details of the accident can be verified in a quick and reliable manner. This will lead to faster delivery of credit to genuine users, declining of credit to false cases and overall there will be considerable monetary benefit to the insurance companies.
2. Theft detection: As the vehicle will be having a Vehicle ID which will update its information via RSU onto the blockchain, we can easily locate the vehicles which are stolen. The design of BCU is such that it serves as a storage for every event and when the stolen vehicle tuned on, its location will reflect on to the blockchain from where we can easily track it.
3. Prediction of false behaviour: Vehicles report information on a periodic basis and by using blockchain we can monitor their behavior over a period of time. By analyzing this information stored on cloud, we can know which nodes have behaved maliciously.
4. Clearing paths for ambulance or other emergency services: When some emergency event happens, it is sensed and updated to blockchain via RSUs. This information can be passed on to other vehicles and RSUs so that they clear the path for ambulance or other vehicles needing assistance.
5. Traffic violation and surveillance information: Cloud offers virtually unlimited storage and by using analysis algorithms we can monitor the traffic violations carried out by users. We can also extract surveillance from the collected information by using data mining or other algorithms.
6. Vehicle Tax and credit: Some countries charge a vehicle tax on vehicles on a yearly basis. By using blockchain, the bills can be directly issued to their accounts. Also credit can be offered on a yearly basis to nodes which report correct information throughout the year.

7 Benefits and challenges

The proposed system paves a way forward to disseminate information with least number of false behaviour nodes. Following are some of the benefits of our proposed system.

1. Security: Blockchain provides enhanced level of security as the data is signed by private key of the user and verified using the corresponding public key. This makes it impossible for the user to deny that they passed the information at later points of time. Moreover, information once mined cannot be changed. To alter the information, one has to change hash value of all the blocks in a limited time which is nearly impossible to achieve.
2. Scalability: As the time interval ‘n’ is calculated on the basis of density of vehicles, the system can easily scale with respect to the number of the vehicles.
3. Accountability: Inability of a user to deny that they passed the information makes this system more accountable to use.
4. Transparency: All of the transactions/ information are public and can be checked or verified by anyone, anytime.

The challenges associated with such a system are as follows:

1. Sybil attack: Sybil attack occurs when an entire group of users become malicious and start reporting false information. These users may be owned by one attacker or multiple attackers.
2. Restricted contact time: Limited window of time to analyze the information by individual BCU of the vehicles.
3. Prediction of behavior: The mobility patterns and behavior of nodes can be easily predicted by analyzing the information stored on blockchain. This is a severe threat as it concerns with personal information of the user.

8 Future work

The proposed system here has covered many of the security related aspects in case of information dissemination in VANETs. However there are some aspects which need further work.

- Designing of an entire unit such as Blockchain Unit as proposed is a rewarding challenge where much more functionality and services can be added to it. It can serve as a black box, similar to the ones found in aircrafts. This can act as a single dedicated information point.
- Also, how the analysis of information will take place at cloud server is a challenge. Defining elaborately about which parameters to sense and carrying out analysis from it is another possible future work direction.
- How to divide the regions amongst a state or country is another question. It can be answered in terms of density of vehicles or roads or events etc. Formulating a complete solution for the same is still needed so that the entire system becomes more integrated.
- Choosing which information dissemination algorithm to use is another potent question. We can deploy different algorithms for different areas and demands.

9 Conclusion

Blockchain has found application in nearly every domain in the past couple of years due to its simple yet profound concept. We used the concept of blockchain to make VANETs more secure in functioning. A blockchain based system for VANETs is proposed which greatly enhances the security of information being exchanged among vehicles. The paper gives a wholesome coverage of the system along with its applications, benefits, and challenges.

References

[1] A. Singhal, Sarishma, and R. Tomar (2017), *Intelligent accident management system using IoT and cloud computing*, Proc. 2nd Int. Conf. Next Gener. Comput. Technol. NGCT 2016, pp. 89–92.

[2] N. Satoshi and S. Nakamoto (2008), *Bitcoin: A Peer-to-Peer Electronic cash system*, Bitcoin, p. 9.

[3] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang (2017), *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends*, Proc. - 2017 IEEE 6th Int. Congr. Big Data, BigData Congr. 2017, pp. 557–564.

[4] R. Shrestha, R. Bajracharya, and S. Y. Nam (2018), *Blockchain-based Message Dissemination in VANET*, Proc. 2018 IEEE 3rd Int. Conf. Comput. Commun. Secur. ICCCS 2018, pp. 161–166.

[5] B. Leiding, P. Memarmoshrefi, and D. Hogrefe (2016), *Self-managed and blockchain-based vehicular ad-hoc networks*, pp. 137–140.

[6] W. Liu, Z. Lu, Z. Liu, Q. Wang, and G. Qu (2018), *A Privacy-Preserving Trust Model Based on Blockchain for VANETs*, IEEE Access, vol. 6, pp. 45655–45664.

[7] P. K. Sharma, S. Y. Moon, and J. H. Park (2017), *Block-VN: A distributed blockchain based vehicular network architecture in smart city*, J. Inf. Process. Syst., vol. 13, no. 1, pp. 184–195.

[8] M. Singh and S. Kim (2018), *Trust Bit: Reward-based intelligent vehicle commination using blockchain paper*, IEEE World Forum Internet Things, WF-IoT 2018 - Proc., vol. 2018–Janua, pp. 62–67.

[9] R. Shrestha, R. Bajracharya, A. P. Shrestha, and S. Y. Nam (2019), *A new-type of blockchain for secure message exchange in VANET*, Digit. Commun. Networks.

[10] K. Christidis and M. Devetsikiotis (2016), *Blockchains and Smart Contracts for the Internet of Things*, IEEE Access, vol. 4, pp. 2292–2303.

[11] F. Schuh and D. Larimer (2017), *Bitshares 2.0: General Overview*, Cryptonomex, vol. 3268, pp. 0–16.

[12] B. Gassend, D. Clarke, M. Van Djik, S. Devadas, and E. Suh (2003), *Caches and merkle trees for efficient memory authentication*, HPCA (High Perform. Comput. Archit. Symp., pp. 1–14.

[13] A. Kiayias, I. Konstantinou, A. Russell, B. David (2016), *A Provably Secure Proof-of-Stake Blockchain Protocol*, IACR Cryptol. …, pp. 1–27.

[14] M. Castro and B. Liskov (1999), *Authenticated Byzantine fault tolerance without public-key cryptography*, System, no. June, pp. 1–12.

[15] D. Schwartz, N. Youngs, and A. Britto (2014), *The Ripple protocol consensus algorithm*, Ripple Labs Inc White Pap., pp. 1–8.

[16] J. Kwon (2014), *TenderMint : Consensus without Mining*, Https://Tendermint.Com/Docs, vol. 6, p. 10.

[17] I. Y. Y. Hsu, M. Wódczak, R. G. White, T. Zhang, and T. R. Hsing (2010), *Challenges, approaches, and solutions in intelligent transportation systems*, ICUFN 2010 - 2nd Int. Conf. Ubiquitous Futur. Networks, pp. 366–371.

ptc® apexada | ptc® objectada®

Complete Ada Solutions for Complex Mission-Critical Systems

- Fast, efficient code generation
- Native or embedded systems deployment
- Support for leading real-time operating systems or bare systems
- Full Ada tasking or deterministic real-time execution

Learn more by visiting: **ptc.com/developer-tools**

ptc

Towards a Formally Verified Space Mission Software Using SPARK

Neto, P., Tojal, J., Veríssimo, J.
Critical Software, SA., Coimbra, Portugal; email: {pmneto, jj-tojal, jverissimo}@criticalsoftware.com
Melo de Sousa, S.
LISP - University of Beira Interior, Covilhã, Portugal; email: desousa@di.ubi.pt

Abstract

This paper discusses the use of formal verification techniques supported by Correct-by-Construction tools for the development of Safety Mission Critical systems in an industrial context. In particular, the document introduces the implementation, verification and validation of the ExoMars Trace Gas Orbiter (TGO) central software that was implemented by Critical Software, SA. in cooperation with Thales Alenia Space for the European Space Agency (ESA) Mars exploration mission. The pre-existing TGO Ada code has been reworked for the need of this project in the SPARK programming language where, along with a strongly typed system, static analysis and deductive verification toolset were used to ensure security and correctness properties.

Keywords: Safety Mission Critical Systems, Formal Methods, Formal Verification, Correct-by-Construction, Design-by-Contract, SPARK.

1 Introduction

A failure in a Safety Mission Critical system can result in loss of life, injury, environmental damage, mission objectives failed, and severe economic losses. There is an increased assurance need of properties such as Reliability, Safety and Security to enhance the system capabilities to prevent such disasters.

Regarding to the Reliability we have of systems, this property is the probability that a system will be operational for a specific period of time, and Safety and Security are directly linked to the reliability that is associated with the implemented system. Safety is a property that ensures that the system will operate flawlessly and Security is the property that states that, despite malfunctions or external threats that may occur, the system is protected and always maintains the proper operating level.

The consequences of the aforementioned failures translate into high costs, increasing the need to introduce new methodologies to assure the critical properties for system reliability. However, these available methodologies are usually discarded when such high quality seal is not required, and, that, for two reasons. The first one is the perception of its cost. Formally ensuring the reliability, the safety and the security of a critical software system is perceived as a time consuming and complex task. The second reason, perhaps the source of the first, is the lack of formal verification experts (or expertise) in usual critical system development teams.

For companies operating in the field of critical systems software development, there is a normative need for the knowledge and the use of techniques for formal code verification and validation such as Model Checking, Theorem Proving and Design-by-Contract. At Critical Software, SA., the most frequently used technique to date for code verification and validation is software testing. A technique that can sometimes become costly due to the late error detection process. The objective of this work is the study of a methodology that can be implemented in the Safety Mission Critical systems development process in order to substantially reduce some of the costs of software testing. The technique to be applied, Design-by-Contract, is a formal code verification methodology usually performed by static analysis and deductive verification tools that will allow the detection of errors during the implementation stage where their correction will be less expensive and, thus partially reducing the costs of software testing activities, in addition to the stronger assurance they provide.

With this goal in mind, the work developed throughout this investigation was to test the SPARK tool commonly used for the formal verification of programs implemented in Ada. The verified code was the central software of ExoMars TGO that is part of an ESA mission to the exploration of the Martian atmosphere where, the main goal is the measurement of its methane levels. ExoMars TGO central software has been implemented in the Ada language by Critical Software, SA. in partnership with Thales Alenia Space.

2 Design-by-Contract

The Design-by-Contract [1] software development paradigm proposes that the implementation should be accompanied by the requirements at the level of each program unit, with a similar granularity to unit testing. These requirements are called contracts and are of two types. A function or code unit has an entry contract and an exit contract. The entry contract (also known as Pre-condition) specifies what conditions (the relation between input data) the code unit expects to meet at the beginning in order to operate as expected. The exit contract (also known as Post-Condition) specifies in which state the program unit leaves the manipulated data (including the returned data) when it returns the hand.

Programming environments that allow the use of this paradigm use formal verification methods that can compare contracts with code and thus define which mathematical properties (essentially logical and arithmetic conditions) must be verified for formal agreement between the contracts and what that programmatic units perform. When these conditions are determined, it is possible to resort to theorem provers who try (as more automatically as possible) to prove or on the contrary refute (with explicit counter example) that these conditions are met. Formally relating Pre-Conditions and Post-Conditions under the action of the code usually hit theoretical limits (decidability issues) and then the whole machinery needs help in the form of extra logical/relational information (referred as invariants, variants or assertions) inserted by the programmer in key places in the code under scrutiny (loops, function calls, non-trivial arithmetic operations, etc.).

For example, consider the code in Listing 2. This exposes only the specification of SPARK contracts for a program that looks for the highest value, for a given position range received by parameter, from an integer vector with a defined range (1 to 1000). We have Min_L and Max_L which are the minimum and maximum limits, respectively, of the required range and Max_V is the returned value.

```
Arr : Int_Array (1 .. 1000);

procedure Max_Val_In(Min_L, Max_L : in Positive;
    Max_V : out Integer)
with
  SPARK_Mode,
  Pre  => Min_L in Arr'Range and Max_L in Arr'Range
          and Min_L <= Max_L,
  Post => (for all I in Min_L .. Max_L => Max_V >= Arr(I))
          and (for some I in Min_L .. Max_L => Max_V = Arr(I)) ;
```

The Pre-Condition specifies that both limits must be within the range set for the vector in order to not attempt invalid positions accesses of the vector. Moreover, to define a valid range the minimum limit must be less than or equal to the maximum limit. The Post-Condition checks whether the returned value is actually the largest integer within that range in the vector and whether it is a value belonging to the vector in the defined range.

3 SPARK

SPARK [2] is a language that allows formal verification of Ada code using the Design-by-Contract approach. The tool allows to annotate the Ada code with the contracts already mentioned in their defined syntax and later uses its verification tool which, with the help of external provers, tries to prove the integrity of the programs. The latest version, SPARK 2014, has integrated a new verification tool that performs static analysis, GNATProve [3], which is aided by Why3 [4] in the programs proof. As depicted in Figure 1, annotations are translated to WhyML and later Proof Obligations are generated which will be proved by external provers such as CVC4 [5], Z3 [6] and Alt-Ergo [7]. SPARK's work is divided into two analyzes, Flow Analysis and Code Verification.

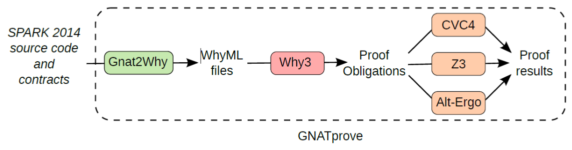


Figure 1: GNATProve Deductive Verification. [8]

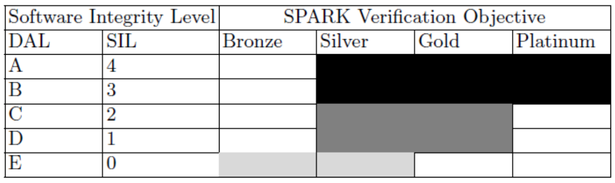
The main concerns of Flow Analysis are data, detecting cases that may raise possible exceptions during execution, such as the use of pointers, alias cases, and side effects to behavior required for functions. Furthermore, it automatically checks if variables and objects are correctly declared, instantiated and updated. It also proves the correct flow of information between programs using special contracts that allow to define the use of global data and their dependencies.

Code Verification is the analysis that proves the integrity of the program. Firstly, SPARK detects the possibility of errors occurring during program execution such as division by zero, overflows, violation of the intervals defined for a given data type, among others. There are two possibilities for the issue of a warning, either the code is really wrong and has to be changed or SPARK has issued a false alarm that can easily be proved by using contracts to set certain properties that are not well known by the tool. Secondly, SPARK uses contracts to prove the functional correctness of the program, that is, whether it behaves according to the conditions specified in the Post-Condition.

The use of SPARK can be divided into 5 levels:

- ***Stone* –** Valid SPARK code;
- ***Bronze* –** Correct data initialization and flow;
- ***Silver* –** Absence of run-time errors;
- ***Gold* –** Proof of critical properties;
- ***Platinum* –** Proof of full functional properties;

According with the results of the last Altran UK industrial projects they summarize in [9], Figure 2, some recommendations of SPARK application compared with different levels of assurance defined by two of the most common software integrity scales (Design Assurance Level (DAL) defined in DO-178B and Software Integrity Level (SIL) defined in IEC 61508).



Software Integrity Level		SPARK Verification Objective			
DAL	SIL	Bronze	Silver	Gold	Platinum
A	4		■	■	■
B	3		■	■	■
C	2		▨	▨	
D	1		▨	▨	
E	0	░	░		

Figure 2: Technical Planning Guidelines for the Application of SPARK. [9]

Note: The required integrity of the higher levels of these scales is assured if there is at least SPARK application at the SILVER level, as we can see in the Black region.

4 ExoMars TGO

In ExoMars, as in all other satellite systems, there are five different systems, as shown in the Mission Application Software layer (Red) of the system architecture, Figure 3. The systems are described as follows:

- **Guidance Navigation Control -** Implements the functions to manage all three modes (Guidance, Navigation, Control);
- **Solar Array Drive Mechanism -** Implements the services that control the motors of the satellite solar panels;
- **Antenna Pointing Mechanism -** Implements the services that control the motors that define the systems antenna position
- **Thermal Regulation -** Implements the services for spacecraft thermal regulation;
- **Entry, Descent Module -** Implements the functions to control the payload that will separate from the spacecraft and land on the surface of Mars;

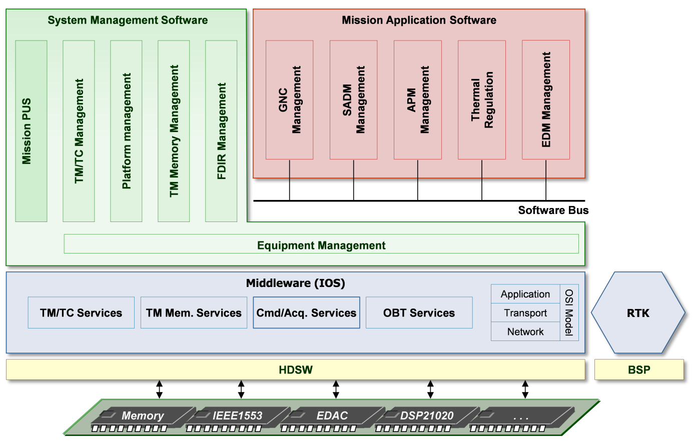


Figure 3: ExoMars TGO Architecture.

For this case study only the temperature regulation system was considered. The TR's function is to compute the spacecraft physical components average temperatures and to activate (or deactivate) heaters to hold the temperature within the recommended thresholds for each hardware component, which can differ between very high or very negative values depending on the orientation of that component relative to the sun or shadow.

5 SPARK Analysis of TR

The TR system performs a cyclic task that constantly regulates the temperature of the hardware components, Figure 4. The cyclic task processing depends on the following main components:

- **Failure Detection, Isolation and Recovery (FDIR)**;
- **Thermal Regulation**;
- **Command List**;

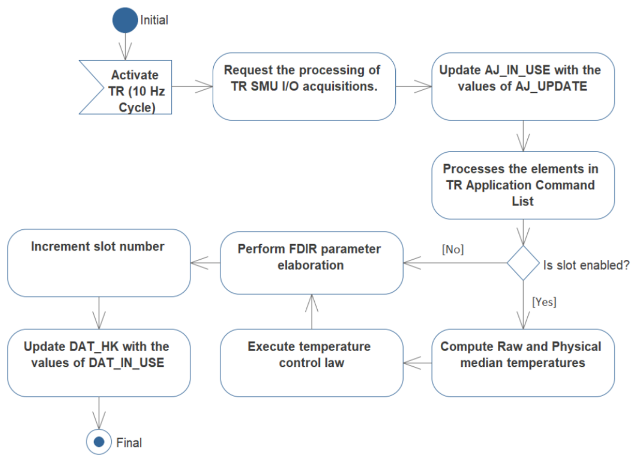

Initial
Activate TR (10 Hz Cycle)
Request the processing of TR SMU I/O acquisitions.
Update AJ_IN_USE with the values of AJ_UPDATE
Processes the elements in TR Application Command List
Is slot enabled?
[No]
[Yes]
Perform FDIR parameter elaboration
Increment slot number
Compute Raw and Physical median temperatures
Execute temperature control law
Update DAT_HK with the values of DAT_IN_USE
Final

Figure 4: TR Cyclic Task.

In order to accomplish the cyclic task SPARK analysis, all of its dependencies full correctness shall be proved beforehand.

As the original code to be analysed was not implemented with SPARK methodologies in mind, some of its implementation features are not supported by the tool. Thus, the viable approach was to retain these features, dissociating from SPARK the functions and/or features not compatible in well marked external interfaces, without influencing the initial behavior of programs.The analysis was then performed in order to ensure the required execution behaviour along with the correct flow of data and the absence of runtime errors. Ultimately, the main goal has always been to achieve SPARK highest level of integrity, Platinum.

6 Results

Stone, Bronze, Silver and Gold levels have been achieved with very positive rates. About the results at the Stone level, one note, that was not possible to perform flow analysis on all system subprograms due to SPARK constraints.

The result of this work ensured the integrity of the system in the Gold level order. The attempt to achieve the maximum SPARK integrity level, Platinum, was not fulfilled. It was not possible to prove the full correctness of the code in the time span the verification team had to do this task. In this time span, the team had to acquire the expertise in SPARK and its verification mechanisms. Another important point is the early use of SPARK in the development process over Ada. In fact, as a result of the following exercise, we have advocated the development team to use several programming styles that ease code modularity and, incidentally, the verification process. Finally, and as already stated, such task requires some expertise of the team and is clearly challenging. Aiming the Platinum level is not recommended when the team is at an early stage of SPARK adoption [10]. Having said that, the full correctness proof of the TR module is, by the opinion of the authors, possible, nothing theoretically challenging where exposed.

From the formal verification effort and extensive code analysis it was also possible to prepare an audit where some minor details were identified that should be considered in future developments, which will help towards a produced code optimization.

7 Conclusions

The verification process described in this document further added a seal of quality to the high levels of confidence that had already been given to the software implemented with the verification and validation process performed by Critical Software and the fact that it has been working very well since system deployment until now. The final results allowed us to conclude that this work has ensured a software quality level equivalent to the required by the highest levels of DO-178B and IEC 61508 standards, according to the table in Figure 2.

The final results of this work increased knowledge to the company through the use of formal code verification techniques such as Design-by-Contract, as well as a set of recommendations to be considered in the process of developing future projects. This project was just the beginning of a journey that encouraged the company to continue with formal methods research. The acquisition of knowledge in this area is important for the company to somehow be able to reduce some costs in software validation and still be prepared for all development phases required by the strictest standards in Safety Mission Critical systems development.

References

[1] B. Meyer, "Applying design-by-contract," *Computer*, vol. 25, pp. 40–51, Oct 1992.

[2] J. W. McCormick and P. C. Chapin, *Building High Integrity Applications with SPARK*. Cambridge University Press, 2015.

[3] J. Guitton, J. Kanig, and Y. Moy, "Why hi-lite ada?," in *Boogie 2011 First International Workshop on Intermediate Language Verification*, p. 27, Citeseer, 2011.

[4] F. Bobot, J.-C. Filliâtre, C. Marché, and A. Paskevich, "Why3 Shepherd your herd of provers," in *Boogie 2011 First International Workshop on Intermediate Verification Languages*, pp. 53–64, 2011.

[5] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli, "Cvc4," in *International Conference on Computer Aided Verification*, pp. 171–177, Springer, 2011.

[6] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, Springer, 2008.

[7] F. Bobot, S. Conchon, E. Contejean, M. Iguernelala, S. Lescuyer, and A. Mebsout, "The alt-ergo automated theorem prover," *http://alt-ergo.lri.fr*, 2008.

[8] D. Hauzar, C. Marché, and Y. Moy, *Counterexamples from proof failures in the SPARK program verifier*. PhD thesis, Inria, 2016.

[9] C. Dross, G. Foliard, T. Jouanny, L. Matias, S. Matthews, J.-M. Mota, Y. Moy, P. Pignard, and R. Soulat, "Climbing the software assurance ladder - practical formal verication for reliable software," in *AVOCS Pre-proceedings*, July 2018.

[10] T. AdaCore, "Implementation guidance for the adoption of spark," 2018.

Ada User Journal

Call for Contributions

Topics: **Ada, Programming Languages**, **Software Engineering Issues** and **Reliable Software Technologies** in general.

Contributions: **Refereed Original Articles**, **Invited Papers**, **Proceedings** of workshops and panels and **News and Information** on Ada and reliable software technologies.

More information available on the
Journal web page at

http://www.ada-europe.org/**auj**

Online archive of past issues at http://www.ada-europe.org/auj/archive/

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o KU Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Hubert B. Keller
Karlsruher Institut für Technologie (KIT)
Institut für Angewandte Informatik (IAI)
Campus Nord, Gebäude 445, Raum 243
Postfach 3640
76021 Karlsruhe
Germany
Email: Hubert.Keller@kit.edu
URL: ada-deutschland.de

Ada-France

attn: J-P Rosen
115, avenue du Maine
75014 Paris
France
URL: www.ada-france.org

Ada-Spain

attn. Sergio Sáez
DISCA-ETSINF-Edificio 1G
Universitat Politècnica de València
Camino de Vera s/n
E46022 Valencia
Spain
Phone: +34-963-877-007, Ext. 75741
Email: ssaez@disca.upv.es
URL: www.adaspain.org

Ada-Switzerland

c/o Ahlan Marriott
Altweg 5
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: president@ada-switzerland.ch
URL: www.ada-switzerland.ch