The journal for the international Ada community

Ada User Journal

Volume 45
Number 4
December 2024

Produced by Ada-Europe

Information on Subscriptions and Advertisements

Ada User Journal (ISSN 1381-6551) is published in one volume of four issues. The Journal is provided free of charge to members of Ada-Europe. Library subscription details can be obtained direct from the Ada-Europe General Secretary (contact details above). Claims for missing issues will be honoured free of charge, if made within three months of the publication date for the issues. Mail order, subscription information and enquiries to the Ada-Europe General Secretary.

For details of advertisement rates please contact the Ada-Europe General Secretary (contact details above).

ADA USER JOURNAL

Volume 45
Number 4
December 2024

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

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

Original Papers

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

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

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

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

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

Proceedings and Special Issues

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

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

News and Product Announcements

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

Reprinted Articles

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

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

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

Editorial

In this final editorial of 2024, I would like to begin by expressing my deepest hope that the foundation of the Ada User Society, earlier this year, will, in a few years, be regarded as a fundamental milestone in the continuity and evolution of the Ada language as an international standard. For this to come true, it will be important that the entire Ada community — including individuals and Ada-related organizations — work together to establish Ada-User as a strong institution with a voice that is heard. In this regard, I would like to highlight the special contribution from the Ada User Society president, Tullio Vardanega, featured on page 219. It emphasizes that "It is Time to Care for Ada."

Concerning the technical contents, we start by providing two articles resulting from presentations at the AEiC 2024 Industrial Track. The first one, by S. Tucker Taft, from AdaCore, describes ongoing work to implement Ada access types and exception handling on top of a ParaSail virtual machine, which does not support user-level pointers and exceptions. The second one, from a group of authors from Spazio IT, GemelliGenerator and Università Cattolica del Sacro Cuore, Italy, looks at the use of generative AI and Large Language Models (LLMs) for software verification. Then, we provide the proceedings of the DeCPS 2024 Workshop, which also took place with AEiC 2024. The reader will find five articles, which present work that was done or is being done in the scope of four research projects (AIDOaRT, Proximity, MYRTUS, and LIONESS). The addressed topics include the use of Natural Language Processing (NLP) techniques and Language Models (LM) for requirements processing in the railway industry, the use of 5G technology to support real-time applications running in the edge, the definition of a model-based system engineering approach to be used in the MYRTUS project, the use of an OpenMP framework to increase resilience and the ability to run applications in heterogeneous architectures, and the use of an evolutionary approach for design space exploration, aiming at improving non-functional properties of cyber-physical systems.

Finally, as usual, we include the *News Digest* section, prepared by its editor Alejandro Mosteo, and the *Calendar and Events* section, prepared by Dirk Craeynest. Please note the announcements of two Ada-related events: the 12th Ada Developer Room at FOSDEM 2025 in Brussels, Belgium, and the 29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025) in Paris, France.

Antonio Casimiro
Lisboa
December 2024
Email: AUJ_Editor@Ada-Europe.org

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/

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

[Messages without subject/newsgroups are replies from the same thread. Messages may have been edited for minor proofreading fixes. Quotations are trimmed where deemed too broad. Sender's signatures are omitted as a general rule. —arm]

Preface by the News Editor

Dear Reader,

Since I am writing this preface in the new year for the first time, let me start by wishing you a prolific 2025. And, staying on the topic of significant dates, I want to highlight an easy-to-miss hidden gem in the thread devoted to the birthday of Lady Ada. There is a simulator of the analytical engine written (in Ada, of course) by long-time Ada practitioner and Ada-on-macOS guru, Simon Wright [1].

This last year also saw the 30th anniversary of the foundation of AdaCore, a key player in the Ada landscape in both private services and open source activities. A number of commemorative communications by the company can be found referenced in this issue [2].

Sincerely,
Alejandro R. Mosteo.

[1] "Happy Birthday, Lady Ada!", in Ada-related Events.

[2] "AdaCore's 30th Anniversary", in References to Publications.

Ada-related Events

Ada Devroom Accepted for FOSDEM 2025

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Subject: [FOSDEM] Ada Devroom accepted for FOSDEM 2025
Date: Wed, 23 Oct 2024 20:00:54 +0200
Newsgroups: comp.lang.ada

Dear Ada users,

The FOSDEM organisation team has accepted the proposal to have an Ada DevRoom for FOSDEM 2025, which will take place in Brussels on the 1st and 2nd of February!

This FOSDEM edition is quite special as it marks its 25th anniversary!

We hope to provide you with more information in the upcoming days, specially with a Call for Presentations.

If someone would like to lend a hand and help organising the DevRoom, please feel free to contact any of us!

Best regards,
The AdaDevroom FOSDEM team. Aka Dirk, AJ & Fernando

Call for Presentations: Ada DevRoom @ FOSDEM 2025

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Subject: [FOSDEM] Call for Presentations: Ada DevRoom @ FOSDEM 2025
Date: Tue, 29 Oct 2024 22:40:26 +0100
Newsgroups: comp.lang.ada

Call for Presentations

12th Ada Developer Room
at FOSDEM 2025

Sunday 2 February 2025,
Brussels, Belgium

www.cs.kuleuven.be/~dirk/
ada-belgium/events/25/
250202-fosdem.html

Organized in cooperation with Ada-Belgium and Ada-Europe

The Ada FOSDEM community is pleased to announce the 12th edition of the Ada DevRoom This edition will take place on Sunday morning 2nd of February in Belgium, at the Université Libre de Bruxelles (ULB). This edition of the Ada DevRoom is once more organized in cooperation with Ada-Belgium [1] and Ada-Europe [2].

General Information about FOSDEM

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 nor payment 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

Ada is a general-purpose programming language originally designed for embedded and mission-critical software engineering, although nowadays it also supports object orientation, contracts and formal verification. It is used extensively in air traffic control, rail transportation, aerospace, nuclear, financial services, medical devices, etc. It is also perfectly suited for open source development with a fully open compiler (part of GCC), a formal verification system and a knowledgeable and vibrant community.

Awareness of safety and security issues in software systems is increasing. The NSA recently published [4] a list of programming languages that are recommended for the development of new software due to their memory safety and Ada was one of the list (one of the three compiled non-garbage collected languages!). In that context, it should be no surprise that NVIDIA has started using Ada/SPARK [5, 6] for their highest critical parts in their GPUs. The forums

[7] have also seen an uptick of new users since the NSA announcement.

Multi-core platforms are now abundant and small, embedded devices are growing exponentially. These are some of the reasons that the Ada programming language and technology attracts more and more attention due to Ada's support for programming by contract, performant and efficient code, high- and low-level abstractions and support for multi-core targets. The latest Ada language definition, Ada 2022, was approved by ISO as an international standard last year. Work on implementing the new features is ongoing, such as improved support for fine-grained parallelism, which were introduced in the new standard. The Ada-related technology, SPARK, provides a complete solution for the safety and security aspects stated above while being fully open source, making it stand out from other formal verification tools, as Ada/SPARK code is compiled directly into ready-to-run programs, which can even run on embedded systems.

More and more tools are available, many are open source, including for small and modern platforms. Interest in Ada keeps increasing, also in the open source community, from which 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 can help 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 [8], a two-day program in 2009 [9], and full day programs in 2012-2016 [10-14], in 2018-2020 [15-17] and 2022 [18]. An important goal is to present exciting Ada technology and projects, including people outside the traditional Ada community. This edition is no different.

Call for Presentations

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

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?

Would you like to share some knowledge and lessons about Ada?

The Ada DevRoom organizers call on you to:

- discuss and help organize the details, subscribe to the Ada-FOSDEM mailing list [19];
- for bonus points, be a speaker: the Ada-FOSDEM mailing list is the place to be!
- don't hesitate to propose a topic that you would like to present to the community, we are eager to know what you have in store for us!

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 should be around 20 or 50 minutes, plus 5 or 10 minutes for Q&A. However, this schedule is flexible and we will adapt it to other formats. For example, a short technical talk can be transformed into a 10 minutes talk, plus time 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 and recorded (audio+video). By submitting a proposal, you agree to being recorded and streamed. You also agree that the contents of your talk will be published under the same license as all FOSDEM content, a Creative Commons (CC-BY) license.

Submission Guidelines

Your proposal must be submitted to the FOSDEM Pretalx system [20]. If you already had an account from previous years, reuse it; if not, create a new account. If, for whatever reason, you cannot use Pretalx, you can also submit your proposal by messaging the Ada-FOSDEM mailing list [15]. If needed, feel free to contact us at the Ada-FOSDEM Mailing list or at <irvise (at) irvise.xyz> (without spaces).

Please, fill the information asked by the Pretalx system, which includes:

- your name, affiliation, contact info;
- the title of your talk (be descriptive and creative);
- a short descriptive and attractive abstract;
- preferred duration of your talk;
- pointers to more information if applicable;
- 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.

Here is the slightly flexible schedule that we will follow:

- November 30, 2024: end of the submission period. Remember, we only need the information in the list above. You do not have to submit the entire talk by this date. Try to submit your proposal as early as possible. It is better to submit half of the details early than be late, so do not wait for the last minute.
- December 15, 2024: announcement of accepted talks.
- January 15, 2025: your slides should be uploaded to the Pretalx platform.
- February 2, 2025: Ada-DevRoom day!

We look forward to lots of feedback and proposals!

Regards,
The Ada-FOSDEM team

Main organiser: Fernando Oleo Blanco <irvise (at) irvise.xyz>

Second in command: Dirk Craeynest <Dirk.Craeynest (at) cs.kuleuven.be>

Third in command: A.J. Ianozi <aj (at) ianozi.com>

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

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

[3] https://fosdem.org

[4] https://www.nsa.gov/Press-Room/News-Highlights/Article/Article/3215760/nsa-releases-guidance-on-how-to-protect-against-software-memory-safety-issues/

[5] https://www.adacore.com/papers/nvidia-adoption-of-spark-new-era-in-security-critical-software-development

[6] https://blog.adacore.com/when-formal-verification-with-spark-is-the-strongest-link

[7] https://forum.ada-lang.io/

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

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

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

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

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

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

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

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

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

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

[18] https://www.cs.kuleuven.be/~dirk/ada-belgium/events/22/220206-fosdem.html

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

[20] https://pretalx.fosdem.org/fosdem-2025/cfp

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Date: Sun, 24 Nov 2024 20:14:52 +0100

As written by Dirk in https://forum.ada-lang.io/t/fosdem-call-for-presentations-ada-devroom-fosdem-2025/1386/12

"""

Proposals are very often submitted very late, which always makes the organisers nervous…;)

That said, we received some proposals by now, but certainly would like to get more. So all of you: if you can tell something about a tool, library, application, technique, case study, experience, etc., that is potentially interesting for FOSDEM, submit a proposal for the Ada DevRoom.

Note that proposals are just that: info about a potential presentation, NOT the presentation itself. There's time to prepare that after the submission deadline. For now, only a small amount of work is needed to submit.

You have until the end of November, less than one week from now. But don't delay: please submit sooner, rather than later.

Dirk
"""

Ada Monthly Meetup, 7th December 2024

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada Monthly Meetup, 7th December 2024
Date: Sat, 16 Nov 2024 10:32:01 -0000
Newsgroups: comp.lang.ada

Original posted on Nov 4, 2024 8:48pm at https://forum.ada-lang.io/t/ada-monthly-meetup-7th-december-2024/ by Fernando aka Irvise <irvise_ml@irvise.xyz>.

Hello everybody!
I would like to announce the December (2024) Ada Monthly Meetup which will be taking place on the 7th of December at 14:00 UTC time (15:00 CET). As always the meetup will take place over at Jitsi. The Meetup will also be livestreamed/recorded to YouTube.

If someone would like to propose a talk or a topic, feel free to do so! We currently have no proposals.

Here are the connection details from previous posts: The meetup will take place over at Jitsi, a conferencing software that runs on any modern browser. The link is Jitsi Meet [1]. The room name is "AdaMonthlyMeetup" and in case it asks for a password, it will be set to "AdaRules". I do not want to set up a password, but in case it is needed, it will be the one above without the quotes. The room name is generally not needed as the link should take you directly there, but I want to write it down just in case someone needs it.

Also, this will be the last Monthly Meetup for a long while! There will be none in January 2025 nor one for February, as a large portion of the Ada community will be meeting in FOSDEM [2]!!

Best regards and see you soon!
Fer

P.S: you can see the October summary in "Ada Monthly Meeting October 2024" [3] or in YouTube [4].

[1] https://meet.jit.si/AdaMonthlyMeetup

[2] https://fosdem.org/2025/

[3] https://forum.ada-lang.io/t/ada-monthly-meetup-5th-october-2024/1209/4

[4] https://www.youtube.com/watch?v=hpbXvSAAu30

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Mon, 9 Dec 2024 13:29:35 -0000

Original posted on Dec 7, 2024 5:12pm at https://forum.ada-lang.io/t/ada-monthly-meetup-7th-december-2024/1444/12 by Fernando aka Irvise <irvise_ml@irvise.xyz>.

The meetup is now over. Here are the minutes of the meetup:

* A strong reminder of the Ada Crate of the Year competition [1]. The deadline is approaching quickly! There are three prices, one for Ada, another for SPARK and finally another for embedded system crates! Here is the forum thread covering the topic [2].
* A reminder that Advent of Code 2024 [3] is now live.
  - AdaCore, like last year, is going to donate money [4] based on the amount of solutions and submission that are being done in Ada or SPARK. The submissions need to be done in this forum thread [5]
* The Learn.AdaCore.com website has received several improvements in these past few months [6]. Some changes are detailed in this blog post [7]. Improvements include controlled and limited types and discriminants.
* The Call for Presentations for the FOSDEM [8] conference has now ended. We did get a nice bunch of submissions which we hope to publish in short time.
* An Ada program was recently showcased in HackerNews [9]. Prunt [10] is a motion controller for 3D printers written in Ada. It is open source, so you can go ahead and take a look at the code.
* There was recently a question in the forum [11] about the use of SPARKlib [12]. The library is a set of nice utilities, algorithms and data structures which have been formally verified. I recommend people to check it out!
* For the people who use AWS [13] in Fedora systems, it is recommended that you read this email thread [14] by Björn Persson [15]. It discusses a security vulnerability disclosed by AdaCore.
* For those interested, charlie5 (Rod Kay) is porting the Linux e1000e network driver to Ironclad. Here is the repo with the progress [16]. Feel free to help and lend a hand. For more info, join Ironclad's Matrix chat room [17].
* The WG9, the ISO Work Group behind the Ada standard, had a meeting a few days ago. Some information [18] was shared with regards to the Ada Users Society. The public list of WG9 documents can be found here [19].

A huge thanks to @AJ-Ianozi [20] for showing us his MMO game whose backend is written using Ada and is based on AWS.

There will be no meetup in January as most people are unavailable on those dates. February will also not have a meetup as we will be in FOSDEM!

Best regards to you all!
Fer

P.S: sorry for the technical issues during the meetup!

[1] https://blog.adacore.com/announcing-the-2024-ada-spark-crate-of-the-year-award

[2] https://forum.ada-lang.io/t/2024-crate-of-the-year-awards/923

[3] https://adventofcode.com/

[4] https://blog.adacore.com/announcing-advent-of-ada-2024-coding-for-a-cause

[5] https://forum.ada-lang.io/t/advent-of-code-2024/1500

[6] https://learn.adacore.com/courses/advanced-ada/changelog.html

[7] https://blog.adacore.com/learn-advanced-ada-2024-09

[8] https://fosdem.org/2025/

[9] https://news.ycombinator.com/item?id=42314905

[10] https://600f3559.prunt-docs.pages.dev/

[11] https://forum.ada-lang.io/t/where-is-sparklib/218/4

[12] https://docs.adacore.com/spark2014-docs/html/ug/en/source/spark_libraries.html#

[13] https://github.com/AdaCore/aws

[14] https://lists.fedoraproject.org/archives/list/ada@lists.fedoraproject.org/thread/IYXTYY2SCZQ32U76MVO5GHK52RXVNQJ6/?noscript

[15] https://lists.fedoraproject.org/archives/users/34271505be0745b085f0fd955fb2e3ef/

[16] https://codeberg.org/charlie5/ironclad_intel_e1000e_driver_port.git

[17] https://matrix.to/#/#ironclad:matrix.org" rel="noopener nofollow ugc

[18] https://www.open-std.org/jtc1/sc22/wg9/n654_WG_9_Future_Plan.pdf

[19] https://www.open-std.org/jtc1/sc22/wg9/documents.htm

[20] https://forum.ada-lang.io/u/aj-ianozi

Happy Birthday, Lady Ada!

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Happy birthday, Lady Ada!
Date: Tue, 10 Dec 2024 16:19:30 -0000
Newsgroups: comp.lang.ada

2024/12/10: birthday of Lady Ada Lovelace, born in 1815, namesake of the #AdaProgramming language.

Happy Programmers' Day!

#AdaEurope #AdaBelgium

https://en.m.wikipedia.org/wiki/Ada_Lovelace

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Tue, 10 Dec 2024 20:59:06 -0000

> 2024/12/10: birthday of Lady Ada Lovelace, born in 1815, namesake of the #AdaProgramming language.

Came across this comic strip <https://sydneypadua.com/2dgoggles/> some years ago: a contrafactual imagining of the adventures of Lovelace and Babbage, getting together to fight crime.

("Crime" being, in Babbage's case, music, and in Lovelace's case, poetry.)

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Wed, 11 Dec 2024 18:18:57 -0000

> Came across this comic strip [...]

Some of the CLA regulars might remember that comic strip. It was mentioned on several occasions here: my CLA archive has postings in 2009, 2011, 2014, and 2023.

Recommended! (I have the book...)
Dirk

From: Simon Wright <simon@pushface.org>
Date: Wed, 11 Dec 2024 19:36:26 +0000

> Recommended! (I have the book...)

I have the first edition... Really good.

For interest, https://github.com/simonjwright/analytical-engine

12th Ada Developer Room at FOSDEM 2025 - Sun 2 Feb - Brussels

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: 12th Ada Developer Room at FOSDEM 2025 - Sun 2 Feb - Brussels
Date: Thu, 19 Dec 2024 17:19:05 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada

[CfPart is included in the Forthcoming Events Section —arm]

Ada-Europe Conference - 2nd Call for Contributions - AEiC 2025

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe Conference - 2nd Call for Contributions - AEiC 2025
Date: Tue, 24 Dec 2024 15:33:00 -0000
Newsgroups: comp.lang.ada,fr.comp.lang.ada,comp.lang.misc

[CfP is included in the Forthcoming Events Section —arm]

Ada-related Resources

[Delta counts are from November 13th to January 29th. —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 29 Jan 2025 19:03 CET
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: _992 (+124) members [1]
- LinkedIn: 3_580 (+31) members [2]
- Stack Overflow: 2_435 (+9) questions [3]
- Ada-lang.io: 321 (+34) users [4]
- Gitter: 277 (+6) people [5]
- Telegram: 214 (+6) users [6]
- Libera.Chat: 78 (+9) concurrent users [7]

[1] https://old.reddit.com/r/ada/

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

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

[4] https://forum.ada-lang.io/u

[5] https://app.gitter.im/#/room/#ada-lang_Lobby:gitter.im

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

[7] https://netsplit.de/channels/details.php?room=%23ada&net=Libera.Chat

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: 29 Jan 2025 19:03 CET
To: Ada User Journal readership

GitHub: >1_000* (=) developers [1]
Rosetta Code: 1_008 (+3) examples [2]
42 (=) developers [3]
Alire: 521 (+39) crates [4]
1_315 (+47) releases [5]
Sourceforge: 242 (-9) projects [6]
Open Hub: 214 (=) projects [7]
Codelabs: 61 (+1) repositories [8]
Bitbucket: 37 (=) repositories [9]

*This number is a lower bound due to GitHub search limitations.

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

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

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

[4] https://alire.ada.dev/crates.html

[5] `alr search --list --full`

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

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

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

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

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: 29 Jan 2025 19:09 CET
To: Ada User Journal readership

[Positive ranking changes mean to go up in the ranking. —arm]

- PYPL Index: 15 (=) 1.21% (+0.08%) [1]

- TIOBE Index: 26 (-1) 0.65% (-0.06%) [2]

- Stack Overflow Survey: 40 (=) 0.9% (=) [3]

- IEEE Spectrum (trending):46 (=) Score: 0.0022 (=) [4]

- IEEE Spectrum (general): 50 (=) Score: 0.0014 (=) [4]

- IEEE Spectrum (jobs): 55 (=) Score: 0.0 (=) [4]

- Languish Trends: 141 (+12) 0.01% (=) [5]

[1] http://pypl.github.io/PYPL.html

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

[3] https://survey.stackoverflow.co/2024/

[4] https://spectrum.ieee.org/top-programming-languages/

[5] https://tjpalmer.github.io/languish/

Re: Adaic.org; Is There a Problem?

[Cont'd from AUJ 45-3, July 2024, this thread discussed the temporary unavailability of the adaic.org website. —arm]

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: adaic.org; is there a problem?
Date: Fri, 1 Nov 2024 22:39:35 -0500
Newsgroups: comp.lang.ada

In case anyone is interested:

AdaIC.org was failing because a PHP update borked the Wordpress "theme" that it uses. We had to pay the hosting company to figure out the problem; they rolled back the PHP and all seems well. This is the bane of using "standard" tools for web hosting; they make it easier to create posts, but also are much more attackable and thus get updated and break a lot more often. (I preferred the old mostly Ada+HTML version, but that wasn't my choice.)

Ada-Auth.Org was down because a water main broke near my office building and flooded the entire lower level with 14" of muddy water. (This of course shorted out most of the electrical things; luckily all of the computers themselves were high enough to escape damage.) All of the tenants had to move out immediately (the building had to be gutted to eliminate mold and rot problems). I ended up moving all of the computers to my living room, and had to get the business internet and phone moved as well. (That was a major mess, still not quite cleaned up.) Everything appears to be working properly now.

Today is the first time I've managed to find time to look at comp.lang.ada since mid-September. Someday I might catch up again, but probably not fully until next year.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sat, 2 Nov 2024 05:34:36 -0000

> they rolled back the PHP and all seems well.

That's not "figuring out" the problem though, is it? The next time you retry the PHP update, won't it just just happen again?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 20 Nov 2024 19:35:07 -0600

My understanding is that the tech tried to fix the problem but couldn't figure it out and gave up and rolled back instead. I agree that isn't a real fix, but that's definitely not something I can do anything about. (I have no interest in figuring out PHP!)

Ada-related Tools

GCC 14.2.0-3 (aarch64, MacOS)

From: Simon Wright <simon@pushface.org>
Subject: [ANN] GCC 14.2.0-3 (aarch64, macOS)
Date: Sun, 17 Nov 2024 14:41:33 +0000
Newsgroups: comp.lang.ada

Available at [1]. Included tools at v25.0.0.

[1] https://github.com/simonjwright/distributing-gcc/releases/tag/gcc-14.2.0-3-aarch64

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Thu, 21 Nov 2024 15:30:22 +0000

Hi Simon, many thanks for that.

I note that 14.2.0-3 has the same issue as all previous 14.2 versions: a spurious error message on a simple string concatenation, which appears only when link-time optimization is called for:

```
gnatmake -aI../Source -aO../Build -funwind-tables -gnatl12j96 -gnatw.e -gnatwD -gnatwH -gnatwP -gnatwT -gnatw.W -gnatw.B -gnatwC -gnatw.u -gnatyO -gnatw.Y -gnatw.N -fdata-sections -ffunction-sections -O3 -flto askance -bargs -static -Sin -largs -Wl,-dead_strip -Wl,-dead_strip -largs -flto

...

gnatbind -aI../Source -aO../Build -static -Sin -x askance.ali
gnatlink askance.ali -funwind-tables -fdata-sections -ffunction-sections -O3 -flto -Wl,-dead_strip -Wl,-dead_strip -flto
lto-wrapper: warning: using serial compilation of 4 LTRANS jobs
lto-wrapper: note: see the '-flto' option documentation for more information
/Users/wf/KDF9/emulation/Source/posix.adb: In function 'posix__output_line':
/Users/wf/KDF9/emulation/Source/posix.adb:277:49: warning: '__builtin_memcpy' writing between 1 and 2147483647 bytes into a region of size 0 [-Wstringop-overflow=]
277 | message_line : constant String :=
    |
message & NL;
          ^
```

This is in the routine:

```
procedure output_line (message: in String)
is
   message_line : String := message & NL;
begin
   output(message_line);
end output_line;
```

where:

```
NL : constant String := OS_specifics.EOL;
```

in the package spec., and:

```
package body OS_specifics is
function EOL
return String
is (1 =>  Character'Val(16#0A#));
```

...

Changing the complained-of line to:

```
message_line : String :=
message &  (NL &  "");
```

makes the warning go away.

From: Moi <findlaybill@blueyonder.co.uk>
Date: Thu, 21 Nov 2024 15:45:53 +0000

I forgot to say that I'm running up-to-date Sequoia.

From: Simon Wright <simon@pushface.org>
Date: Thu, 21 Nov 2024 20:40:18 +0000

I can't reproduce.

The only change made to the 14.2.0 compiler release are to do with fixincludes vs SDK16.

-1: gcc-14.2-darwin-r1
-2, -3: gcc-14.2-darwin-r2

From: Moi <findlaybill@blueyonder.co.uk>
Date: Fri, 22 Nov 2024 01:18:36 +0000

> I can't reproduce.

I'm not surprised, it seems to be very dependent on the exact context. I could not reproduce it in a simple example. It has been present in all 14* compilers, but it is absent from 13.2.0.

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>
Date: Fri, 22 Nov 2024 11:02:23 +0100

> warning: '__builtin_memcpy' writing between 1 and 2147483647 bytes into a region of size 0 [-Wstringop-overflow=]

I've seen many occurrences of that bogus warning in Fedora, so it's not unique to Simon's build. Thus the place to report it would be https://gcc.gnu.org/bugzilla/.

From: Moi <findlaybill@blueyonder.co.uk>
Date: Sun, 24 Nov 2024 00:19:06 +0000

Do you get it only in 14.2.0-*, Björn?

From: Simon Wright <simon@pushface.org>
Date: Sun, 24 Nov 2024 11:17:07 +0000

It happens with GCC version 15.0.0 20241102 (experimental) :-(

And with an x86_64-apple-darwin GCC 4.2.0 build (on aarch64, under Rosetta) I noticed

```
In function
'disassembly__data_access_name',
inlined from
'disassembly__the_full_name_of' at
/Users/simon/tmp/emulation/Source/
disassembly.adb:483:44:
/Users/simon/tmp/emulation/Source/disas
sembly.adb:452:48: warning:
'__builtin_memcpy' writing between 1
and 2147483647 bytes into a region of
size 0 overflows the destination
[-Wstringop-overflow=]
452 | modifier  : constant String  :=
      |
M_suffix & Q_suffix;
           ^
/Users/simon/tmp/emulation/Source/
disassembly.adb:452:48: note: destination
object 'S1471b.315' of size 0
452 | modifier  : constant String  :=
      |
M_suffix & Q_suffix;
           ^
/Users/simon/tmp/emulation/Source/disas
sembly.adb:452:48: warning:
'__builtin_memcpy' writing between 1
and 2147483647 bytes into a region of
size 0 overflows the destination [-
Wstringop-overflow=]
452 | modifier  : constant String  :=
      |
M_suffix & Q_suffix;
           ^
/Users/simon/tmp/emulation/Source/disas
sembly.adb:452:48: note: destination
object 'S1471b.315' of size 0
452 | modifier  : constant String  :=
      |
M_suffix & Q_suffix;
           ^
```

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>
Date: Tue, 26 Nov 2024 16:48:31 +0100

> Do you get it only in 14.2.0-*, Björn?

I haven't taken notes of the GCC version. I just concluded that the warning was obviously wrong, and moved on. I can let you know if I see it again.

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>
Date: Fri, 29 Nov 2024 20:08:53 +0100

> Do you get it only in 14.2.0-*, Björn?

It happened today in GCC 14.2.1 (as packaged in Fedora 41), so no, not only in 14.2.0.

From: Simon Wright <simon@pushface.org>
Date: Fri, 29 Nov 2024 20:52:08 +0000

> It happened today in GCC 14.2.1 (as packaged in Fedora 41), so no, not only in 14.2.0.

There's no official FSF 14.2.1 release - it may just be like Alire, which only handles 3 levels, so they call the first packaging of 14.2.0 14.2.1. If you say 'gcc -v' it'll probably say 14.2.0.

Of course I could be completely wrong and Fedora have added lots of value!

From: Keith Thompson <keith.s.thompson+u@gmail.com>
Date: Fri, 29 Nov 2024 13:21:30 -0800

> There's no official FSF 14.2.1 release - it may just be like Alire, which only handles 3 levels, so they call the first packaging of 14.2.0 14.2.1

I see some potential for confusion, since there almost certainly will be an official GCC 14.2.1 release in the near future. That official release will include code that's not included in what Fedora calls gcc 14.2.1. It's a point release, so I wouldn't expect substantial changes, but still, I think Fedora should use a different naming scheme.

From: Simon Wright <simon@pushface.org>
Date: Fri, 29 Nov 2024 22:25:22 +0000

> I see some potential for confusion, since there almost certainly will be an official GCC 14.2.1 release in the near future.

I have this vague memory that 14.2.1 is a draft for 14.3.0, and that GCC never has .1 _releases_. In the same way that 15.0.0 migrates to 15.0.1 before the 15.1.0 release.

Yes, see [1], section "Version Numbering Scheme for GCC 5 and Up".

Also, in gcc-mirror on github, gcc/BASE-VER in the releases/gcc-14 branch [2] holds 14.2.1.

[1] https://gcc.gnu.org/develop.html

[2] https://github.com/gcc-mirror/gcc/blob/releases/gcc-14/gcc/BASE-VER

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>
Date: Fri, 29 Nov 2024 23:52:17 +0100

> If you say 'gcc -v' it'll probably say 14.2.0.

```
$ LANG=en GCC -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexe
c/gcc/x86_64-redhat-linux/14/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-
none:amdgcn-amdhsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-redhat-linux
Configured with: ../configure --enable-
bootstrap --enable-
languages=c,c++,fortran,objc,obj-
c++,Ada,go,d,m2,lto --prefix=/usr --
mandir=/usr/share/man --
infodir=/usr/share/info --with-
bugurl=http://bugzilla.redhat.com/bugzilla
--enable-shared --enable-threads=posix --
enable-checking=release --enable-multilib
--with-system-zlib --enable-__cxa_atexit -
-disable-libunwind-exceptions --enable-
gnu-unique-object --enable-linker-build-
id --with-gcc-major-version-only --
enable-libstdcxx-backtrace --with-
libstdcxx-zoneinfo=/usr/share/zoneinfo --
with-linker-hash-style=gnu --enable-
plugin --enable-initfini-array --with-
isl=/builddir/build/BUILD/gcc-14.2.1-
build/gcc-14.2.1-20240912/obj-x86_64-
redhat-linux/isl-install --enable-offload-
targets=nvptx-none,amdgcn-amdhsa --
enable-offload-defaulted --without-cuda-
driver --enable-gnu-indirect-function --
enable-cet --with-tune=generic --with-
arch_32=i686 --build=x86_64-redhat-
linux --with-build-config=bootstrap-lto --
enable-link-serialization=1
Thread model: posix
Supported LTO compression algorithms:
zlib zstd
gcc version 14.2.1 20240912 (Red Hat
14.2.1-3) (GCC)
```

> I see some potential for confusion, since there almost certainly will be an official GCC 14.2.1 release in the near future. That official release will include code that's not included in what Fedora calls gcc 14.2.1. It's a point release, so I wouldn't expect substantial changes, but still, I think Fedora should use a different naming scheme.

Anyone who needs to know the exact state of the code packaged in Fedora 41 can find the Git revision and all the patches here: https://src.fedoraproject.org/rpms/gcc/blob/f41/f/gcc.spec

If someone has a problem with the version numbering, they could always try

reporting it as a bug, but they should first know that the main admin of the Fedora package is also frequently seen in the revision history at gcc.gnu.org. One who is that deeply involved in the development of GCC probably has a better idea of the differences between versions than most of us do.

From: Keith Thompson <keith.s.thompson+u@gmail.com>
Date: Sat, 30 Nov 2024 13:15:24 -0800

After I wrote the above, I built GCC from its git repo, using the tip of the releases/gcc-14 branch (not a release tag). Presumably the Fedora folks did something similar. The resulting GCC reports its own version as:

$ gcc –version
gcc (GCC) 14.2.1 20241129

whereas a gcc built from the releases/gcc-14.2.0 tag reports:

$ gcc –version
gcc (GCC) 14.2.0

with no date.

I still see some potential for confusion, but not as much as I initially thought. (The fact that GCC doesn't do *.*.1 releases is fairly obscure, and I wouldn't expect most people to know about it.)

The PragmAda Reusable Components

From: Pragmada Software Engineering <pragmada@pragmada.x10hosting.com>
Subject: [Reminder] The PragmAda Reusable Components
Date: Sun, 1 Dec 2024 11:21:34 +0100
Newsgroups: comp.lang.ada

The PragmARCs are a library of (mostly) useful Ada reusable components provided as source code under the GMGPL or BSD 3-Clause license at https://github.com/jrcarter/PragmARC.

This reminder will be posted about every six months so that newcomers become aware of the PragmARCs. I presume that those who want notification when the PragmARCs are updated have used Github's notification mechanism to receive them, so I no longer post update announcements. Anyone who wants to receive notifications without using Github's mechanism should contact me directly.

Vulnerability in AWS.Client

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>
Subject: Anyone using AWS.Client in Fedora? You need Rawhide.
Date: Fri, 6 Dec 2024 19:45:39 +0100
Newsgroups: comp.lang.ada

Anyone who uses the client-side HTTPS functionality of the Ada Web Server library needs to know about CVE-2024-37015. HTTPS requests made with AWS.Client are vulnerable to monster-in-the-middle attacks.

Here's the announcement from AdaCore: https://docs.adacore.com/corp/security-advisories/SEC.AWS-0031-v2.pdf

Although the vulnerability was disclosed in August, version 25.0.0 is the only public release that includes the fix. It is now finally available in Fedora, but only in Rawhide, the development version that will become Fedora 42.

The fix comes with API changes that make it difficult to backport to older versions. That also means that programs using AWS will probably need to be adapted to use version 25. Furthermore, AWS 25 needs Gnatcoll 25, and as usual each new library version has a new soname. If we would push AWS 25 and Gnatcoll 25 as updates to Fedora 40 and 41, then any programs using Gnatcoll would stop working when users install the update, even if they have nothing to do with AWS. That would be bad.

Thus, AWS.Client in Fedora 40 and 41 should not be used except on isolated networks where everything on the network is fully trusted. Only in Rawhide is AWS.Client suitable for use on the Internet.

If you run programs in Fedora that use AWS.Client on the Internet, these are your options:

1: Install Rawhide and follow the development version, accepting the instability and the higher maintenance burden, until Fedora 42 is released. Adapt your programs to the API changes in AWS 25. Recompile more or less all of your own programs. Expect further recompilations before the release date, such as when the soname of Libgnat will change sometime in January.

2: Download the source RPM packages of AWS 25 and Gnatcoll 25 from Rawhide, and compile them yourself on Fedora 41. Adapt your programs to the API changes, and also recompile anything that uses Gnatcoll.

This situation is not how I wish it were, but there are limits to what packagers can do when the upstream developers don't make clean bugfix releases.

From: Niocláiſín Cóilín De Ġloſtéir <master_fontaine_is_dishonest@strand_in_london.gov.uk>
Date: Fri, 6 Dec 2024 20:01:49 +0100

"[. . .]

Furthermore, AWS 25 needs Gnatcoll 25, and as usual each new library version has a new soname. If we would push AWS 25 and Gnatcoll 25 as updates to Fedora 40 and 41, then any programs using Gnatcoll would stop working when users install the update, even if they have nothing to do with AWS. That would be bad."

An electronic engineer who also used to be a GNU/Linux administrator had already perceived this problem of shared libraries in the Year 2006.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Fri, 6 Dec 2024 20:15:29 -0000

> An electronic engineer who also used to be a GNU/Linux administrator had already perceived this problem of shared libraries in the Year 2006.

Note the soname part. That only increments when there are backward-incompatible changes to the ABI. Otherwise you could install the new version, and existing compiled code would run against it just fine.

Consider the transition in the Linux world from libc5 to GNU libc6. That happened over 20 years ago. Isn’t time for libc7? No, because they have been able to keep all the changes backward-ABI-compatible so far.

References to Publications

AdaCore’s 30th Anniversary

From: Alejandro R. Mosteo
Subject: AdaCore’s 30th anniversary
Date: Wed, 29 Jan 2025 21:06 +0100
To: Ada User Journal readership

AdaCore, one of the more prominent companies in the Ada world, has celebrated its 30th anniversary in 2024. A number of blog entries have been published on its website; since these are rather long, I include here the references and a brief excerpt. Interested readers can find the complete entries at the company’s website by following the provided links.

July 15, 2024: Celebrating 30 years of AdaCore [1]:

This press release revisits AdaCore's 30-year legacy of tools for reliable, safe, and secure software, highlighting its pioneering work with the Ada language, open-source commitment, and expansion into languages like Rust and technologies like SPARK/Ada.

October 8, 2024: AdaCore Memories: the stories behind the first 30 years of AdaCore [2]:

This blog entry presents an interview with co-founders Ed Schonberg and Richard Kenner, in which they reminisce about the beginnings of the company.

December 11, 2024: AdaCore's 2024; Highlights from our 30th year in Business [3]:

This blog post provides an overview of AdaCore's activities in 2024, during its 30th anniversary, including some key projects and the release of the GNAT Pro 25 toolsuite.

[1] https://www.adacore.com/press/celebrating-30-years-of-adacore

[2] https://blog.adacore.com/adacore-memories-the-stories-behind-the-first-30-years-of-adacore

[3] https://blog.adacore.com/adacores-2024-highlights-from-our-30th-year-in-business

Ada Practice

'Reduce from Right to Left

From: b.mcguinness747@gmail.com (Brian9000)
Subject: How can I make Reduce run from right to left?
Date: Tue, 10 Dec 2024 17:20:53 +0000
Newsgroups: comp.lang.ada

I am trying to find a way to make Reduce operate from right to left like APL reduction, but this doesn't seem to work.

I wrote a test program:

```
pragma Ada_2022;
pragma Extensions_Allowed (On);

with Ada.Text_IO;

procedure Reverse_Reduction is
  type Real is digits 15;
  type Vector is array(Natural range <>)
  of Real;

  data : constant Vector := [for i in 0 .. 11 =>
  Real (i + 1)];
begin
  for i in data'Range loop
     Ada.Text_IO.Put_Line (i'Image & ": " &
     data(i)'Image);
  end loop;
  Ada.Text_IO.New_Line;
  Ada.Text_IO.Put_Line (Real'Image
   ([ for i in 1 .. data'Last => data(i)
   ]'Reduce("-", data(0))));
  Ada.Text_IO.Put_Line (Real'Image
  ([ for i in reverse 0 .. data'Last -
   1 => data(i) ]'Reduce
    ("-", data(data'Last))));
end Reverse_Reduction;
```

but the compiler won't accept "reverse" in the next-to-last line, even though you can use it in a normal loop. The other lines compile and run without any problems.

```
$ gnatmake reverse_reduction.adb
x86_64-linux-gnu-gcc-12 -c
reverse_reduction.adb
reverse_reduction.adb:18:47: error: missing
operand
x86_64-linux-gnu-gnatmake-12:
"reverse_reduction.adb" compilation error
```

Of course I can just write a normal loop to do this, but I wanted to see if I could do it with Reduce.

From: J-P. Rosen <rosen@adalog.fr>
Date: Tue, 10 Dec 2024 18:54:27 +0100

ARM 2022/4.5.10(6/5):
The iterated_element_association of a value_sequence shall not have a key_expression, nor shall it have a loop_parameter_specification that has the reserved word reverse.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Tue, 10 Dec 2024 22:04:02 +0100

> Ada.Text_IO.Put_Line (Real'Image ([for i in reverse 0 .. data'Last - 1 => data(i)]'Reduce("-", data(data'Last))));

Presumably you could do

```
[for I in 0 .. Data'Last - 1 =>
Data (Data'Last - 1 - I)]'Reduce
```

Or write a Reverse function

```
Reverse (Data (0 .. Data'Last - 1) )'Reduce
```

From: b.mcguinness747@gmail.com (Brian9000)
Date: Wed, 11 Dec 2024 01:38:59 +0000

I finally got an APL-like reduction using Ada Reduce:

```
pragma Ada_2022;
pragma Extensions_Allowed (On);

with Ada.Text_IO;

procedure Reverse_Reduction is
  type Real is digits 15;
  type Vector is array(Natural range <>)
  of Real;

  data : constant Vector := [for i in 0 .. 11 =>
  Real (i + 1)];

  function Minus (Accumulator : Real;
   Value : Real) return Real is
   (Value –  Accumulator);
begin
  for i in data'Range loop
     Ada.Text_IO.Put_Line (i'Image & ": " &
     data(i)'Image);
  end loop;
  Ada.Text_IO.New_Line;
  Ada.Text_IO.Put_Line (Real'Image
   ([ for i in 0 .. data'Last - 1 =>
   data(data'Last - 1 - i) ]'Reduce(Minus,
   data(data'Last))));
end Reverse_Reduction;
```

```
$ ./reverse_reduction
0:  1.00000000000000E+00
1:  2.00000000000000E+00
2:  3.00000000000000E+00
3:  4.00000000000000E+00
4:  5.00000000000000E+00
5:  6.00000000000000E+00
6:  7.00000000000000E+00
7:  8.00000000000000E+00
8:  9.00000000000000E+00
9:  1.00000000000000E+01
10:  1.10000000000000E+01
11:  1.20000000000000E+01

-6.00000000000000E+00
```

In GNU APL:

```
  ⍳12
1 2 3 4 5 6 7 8 9 10 11 12
  -/⍳12
¯6
```

Re: Ichbiah 2022 Compiler Mode

[Cont'd from AUJ 45-3, July 2024, this thread discussed hypothetical enhancements Jean Ichbiah might have wanted to see on modern Ada revisions. —arm]

From: Lioneldraghi <lionel.draghi@free.fr>
Subject: Re: Ichbiah 2022 compiler mode
Date: Sat, 21 Dec 2024 00:26:06 +0100
Newsgroups: comp.lang.ada

I am sensible to the complexity vs use balance, and I would easily cope with the negative consequences of most of the simplifications you propose (at least those I understand!) but obviously it's hard to be fully aware of the consequences.

> (3) A number of syntax options are eliminated. Matching identifiers are required at the end of subprograms and packages. Initializers are always required (<> can be used if default initialization is needed). Keyword "variable" is needed to declare variables (we do not want the worst option to be the easiest to write, as it is in Ada).

Why are you considering variables worse than constants?

I don't want the "worst" option to be the easiest to write, but neither do I want to put one more keyword in the most common case.

Note that:

1. I have no statistics, but it seems to me that there are more variables than constants in my code.

2. I say "Useless" from my coder point of view, I don't know if it simplifies the work for compiler or tools implementers.

From: Lioneldraghi <lionel.draghi@free.fr>
Date: Sat, 21 Dec 2024 01:52:43 +0100

> (10) Variable-returning functions are introduced. They're pretty similar to the semantics of anonymous access returns (or the aliased function returns suggested by Tucker). This means that a variable can easily be treated as a

> function (and indeed, a variable declaration is just syntactic sugar for such a function).

I suppose that to allows the compiler to discriminate this nonsense code

> Square (2) := 3;

from the legitimate

> List.Index (3) := Item;

you will have to introduce some specific syntax, like Tucker's "aliased function".

I see the huge benefit from a user point of view, but I'm not aware of compiler internals: doesn't the introduction of a second function type increase complexity?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 21 Dec 2024 02:14:00 -0600

> Why are you considering variables worse than constants? I don't want the "worst" option to be the easiest to write, but neither do I want to put one more keyword in the most common case.

A lot of "variables" in code actually are only written once. In Ada, those are better modeled as constants. A constant tells the reader that the value doesn't change during the life of the object, which is easier for analysis (both human and machine).

Secondly, I am assuming that automation is helping to write a lot of code. "One more keyword" is irrelevant in terms of creating code, the only question is whether it hurts readability. I prefer to have most things written explicitly (but not to the point of noise, of course). That seems especially true if the code is being written by a program and mostly you are trying to figure out why it doesn't work!

> Note that:

> 1. I have no statistics, but it seems to me that there is more variables than constants in my code.

But how many of them *have* to be variables vs. the number that just are because it is easier? I know I have a number of the latter.

> 2. I say "Useless" from my coder point of view, I don't know if it simplifies the work for compiler or tools implementers.

Constants do help the compiler generate better code, although a lot of the benefits can be gained also by working harder. (That's what C compilers do, after all.)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 21 Dec 2024 02:19:09 -0600

>> (10) Variable-returning functions are introduced. [...]

> I suppose that to allows the compiler to discriminate this nonsense code

> Square (2) := 3;

> from the legitimate

> List.Index (3) := Item;

> you will have to introduce some specific syntax, like Tucker's "aliased function".

> I see the huge benefit from a user point of view, but I'm not aware of compiler internals: doesn't the introduction of a second function type increase complexity?

Yes, but Ada already has a bunch of different mechanisms for dealing with objects/functions/exceptions/packages/types. My intent is to collapse those all into one (somewhat more complex) mechanism. The basic idea is that everything resolves a single way, meaning that everything can be overloaded, and there no longer is a semantic difference between:

```
A : constant T := ...;
```

and

```
function A return T is (...);
```

Whether that really helps remains to be seen, of course. But the goal is to reduce the number of disjoint mechanisms both in the language description and in the implementation. The hope is then to be able to introduce additional capabilities on top of a simpler and stronger foundation.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 21 Dec 2024 10:50:41 +0100

> "One more keyword" is irrelevant in terms of creating code, the only question is whether it hurts readability.

More keywords = fewer words that can be used as identifiers so more keywords make writing code a little bit harder than fewer keywords.

> But how many of them *have* to be variables vs. the number that just are because it is easier? I know I have a number of the latter.

I put a lot of effort into making sure that all constants are so declared, because I have the rule that (with certain exceptions) no non-local variables may be referenced from subprograms, but constants may be referenced from anywhere. However, I sometimes have constants that cannot be initialized with a single expression, resulting in

```
C : T; -- Constant after initialization
```

Once C has been initialized, I treat it as a constant. Would your approach allow the compiler to know that C is really a constant?

From: G.B. <bauhaus@notmyhomepage.invalid>
Date: Sat, 21 Dec 2024 18:19:12 +0100

> Constants do help the compiler generate better code, although a lot of the benefits can be gained also by working harder. (That's what C compilers do, after all.)

What are some compilers offering today? That is, can they find declarations of variables that could be constants, if so instructed?

I am seeing some warnings about non-initialized variables for a meaningless mock-up, but not much else. Ada, C++, Java.

(Maybe there are options that I have missed. Or an analysis of a whole program yields more.)

```
function testc (b : Boolean) return Integer is

package P is
   x : Integer;
end;

begin
   if b then
            P.x := 42;
   end if;
   return P.x;
end testc;
```

```
int testc(bool b) {
    struct {
       int x;
   } P;
   if (b) {
      P.x = 42;
   }
   return P.x;
}
```

```
class testc {
   class P {
      int x;
    }
   P P;
   int $(boolean b) {
      if (b) {
         P.x = 42;
       }
       return P.x;
   }
}
```

From: Chris Townley <news@cct-net.co.uk>
Date: Sat, 21 Dec 2024 17:35:30 +0000

> [Code sample from previous post. – arm]

My understanding is that many compilers will optimise these, and if trivial numbers will 'optimise' out the variable. Confusing in the debugger!

From: Keith Thompson <keith.s.thompson+u@gmail.com>
Date: Sat, 21 Dec 2024 13:26:06 -0800

> A lot of "variables" in code actually are only written once. In Ada, those are better modeled as constants. A constant tells the reader that the value doesn't

change during the life of the object, which is easier for analysis (both human and machine).

Agreed. But my understanding is that compilers typically do this kind of analysis anyway, at least when optimization is enabled. For example, if I write:

```
N : Integer := 42;
```

and later refer to the value of N, if the compiler is able to prove to itself that N is never modified after its initialization, it can replace a reference to N with the constant 42 (and possibly fold it into other constant expressions).

Using "constant" for something that isn't going to be modified is good practice, but I'd say it's for the benefit of the human reader.

From: Pascal Obry <pascal@obry.net>
Date: Sun, 22 Dec 2024 10:32:20 +0100

> Using "constant" for something that isn't going to be modified is good practice, but I'd say it's for the benefit of the human reader.

Exactly the point of Randy. And as Jean-Pierre would say "a program is written once and read many times" (or something like that).

So having "constant" is a very important point when reading the code for maintenance. Actually a developer doesn't care about the compiler, what is important is the "visible" semantic to the human reader.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 23 Dec 2024 19:00:32 -0600

> I put a lot of effort into making sure that all constants are so declared, because I have the rule that (with certain exceptions) no non-local variables may be referenced from subprograms, but constants may be referenced from anywhere.

Precisely. The idea is to encourage use of constants by eliminating the unnatural advantage to writing uninitialized variables. If everything is equally easy/hard to write, then one is more likely to make the best choice for the program.

> However, I sometimes have constants that cannot be initialized with a single expression, resulting in

> C : T; -- Constant after initialization

> Once C has been initialized, I treat it as a constant. Would your approach allow the compiler to know that C is really a constant?

Not with the approach I was envisioning. Of course, Ada 2022 and beyond already make it possible to initialize a lot more objects (especially with the introduction of container aggregates), so hopefully it will be less necessary to write things like your example.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 23 Dec 2024 19:03:53 -0600

> What are some compilers offering today? That is, can they find declarations of variables that could be constants, if so instructed?

I was thinking from a code generation perspective, as opposed to static analysis. These are really the same process, but in one case the output is only for a computer, and in the other, the output is for a human. The needs of those outputs are quite different. I think a lot of compilers do optimizations of constants by discovery, but few compilers do much beyond rudimentary static analysis. I expect that to change, but I may be an optimist on 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.

2025

☺ January 19-25 52nd ACM SIGPLAN **Symposium on Principles of Programming Languages** (POPL'2025), Denver, Colorado, USA. Topics include: all aspects of programming languages and programming systems, both theoretical and practical; fundamental principles and important innovations in the design, definition, analysis, transformation, implementation and verification of programming languages, programming systems, and programming abstractions.

January 20-21 **International Conference on Certified Programs and Proofs** (CPP'2025). Topics include: research areas related to formal certification of programs and proofs; new languages and tools for certified programming; program analysis, program verification, and program synthesis; program logics, type systems, and semantics for certified code; teaching mathematics and computer science with proof assistants; etc.

January 20-21 26th **International Conference on Verification, Model Checking, and Abstract Interpretation** (VMCAI'2025), Denver, Colorado, USA. Co-located with POPL'2025. Topics include: program verification, model checking, abstract interpretation, static analysis, type systems, program certification, detection of bugs and security vulnerabilities, hybrid and cyber-physical systems, concurrent and distributed systems, analysis of numerical properties, analysis of smart contracts, etc., case studies on all of the above topics.

January 20-22 20th **International Conference on High Performance and Embedded Architecture and Compilation** (HiPEAC'2025), Barcelona, Spain. Topics include: computer architecture, programming models, compilers and operating systems for general-purpose, embedded and cyber-physical systems.

☺ January 20 6th **Workshop on Next Generation Real-Time Embedded Systems** (NG-RES'2025). Topics include: application of formal methods to distributed and/or parallel real-time systems; programming models, paradigms and frameworks for real-time computation on parallel and heterogeneous architectures; dependable systems and networks; compiler-assisted solutions for distributed and/or parallel real-time systems; middlewares for distributed and/or parallel real-time systems; scheduling and schedulability analysis for distributed and/or parallel real-time systems; etc.

♦ February 02 12th **Ada Developer Room at FOSDEM 2025**, Brussels, Belgium. FOSDEM 2025 is a two-day event (Sat-Sun 1-2 Feb), held in hybrid mode. This year's edition includes once more an Ada Developer Room, held on Sunday morning 2 February, and organized in cooperation with Ada-Belgium and Ada-Europe.

February 04-06 19th **International Working Conference on Variability Modelling of Software-Intensive Systems** (VaMoS'2025), Rennes, France. Topics include: variability across the software lifecycle, test and verification of variable systems, evolution of variability-intensive systems, runtime variability, variability mining, reverse-engineering of variability, economic aspects of variability, variability and quality requirements, industrial development of variable systems, experience reports from managing variability in practice, etc.

February 24-28 **Software Engineering 2025** (SE'2025), Karlsruhe, Germany. Event includes workshops: Automotive Software Engineering (ASE'25), 7th Workshop on Avionics Systems and Software Engineering

(AvioSE'25), 9th Workshop on Software Engineering for Cyber-Physical Production Systems (SECPPS'25), etc.

March 01-05 ACM/IEEE **International Symposium on Code Generation and Optimization** (CGO'2025), Las Vegas, USA.

March 04-07 32nd IEEE **International Conference on Software Analysis, Evolution, and Reengineering** (SANER'2025), Montreal, Quebec, Canada. Topics include: software tools for software evolution and maintenance; software analysis, parsing, and fact extraction; software reverse engineering and reengineering; program comprehension; software evolution analysis; software architecture recovery and reverse architecting; program transformation and refactoring; mining software repositories and software analytics; software reconstruction and migration; software maintenance and evolution; program repair; software release engineering, continuous integration and delivery; empirical studies on all the above topics; education related to all of the above topics; etc.

March 12-14 33rd Euromicro/IEEE **International Conference on Parallel, Distributed and Network-Based Processing** (PDP'2025), Turin, Italy. Topics include: embedded parallel systems; dependability, survivability, fault-tolerance; programming languages, compilers, middleware; runtime, systems software; performance prediction and analysis; simulation and modeling of parallel/distributed systems; etc.

Mar 30 - Apr 03 20th **European Conference on Computer Systems** (EuroSys'2025), Rotterdam, the Netherlands. Topics include: all areas of computer systems research, such as distributed systems, language support and runtime systems, systems security and privacy, dependable systems, analysis, testing and verification of systems, parallelism, concurrency, and multicore systems, real-time, embedded, and cyber-physical systems, etc.

Mar 31 - Apr 04 40th ACM/SIGAPP **Symposium on Applied Computing** (SAC'2025), Catania, Italy. Topics include: latest developments, trends, experiences, and challenges in applied computing. 42 specialized tracks, including: cyber-physical systems; dependable, adaptive, and secure distributed systems; embedded system; IoT and edge computing; interoperability; programming languages; software engineering; computer security; software verification and testing; etc.

☺ Mar 31-Apr 04 **Track on Programming Languages** (PL'2025). 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. Deadline for submissions: January 9, 2025 (student travel award program application).

Mar 31-Apr 04 **Software Verification and Testing Track** (SVT'2025). 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, abstract interpretation, 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 31-Apr 04 20th **Track on Dependable, Adaptive, and Secure Distributed Systems** (DADS'2025). Topics include: Dependable, Adaptive, and secure Distributed Systems (DADS); modeling, design, and engineering of DADS; foundations and formal methods for DADS; applications of DADS; etc.

Mar 31 - Apr 04 22nd IEEE **International Conference on Software Architecture** (ICSA'2025), Odense, Denmark. Topics include: linking architecture to requirements and/or implementation; methods to address the intertwining of specification and design; model-driven architecture; component-based software engineering; architecture frameworks and architecture description languages; evaluating quality aspects (e.g., security, performance, reliability, evolvability); automatic extraction and generation of software architecture descriptions; architecture & continuous integration/delivery, and DevOps; refactoring and evolving architecture design decisions and solutions; roles and responsibilities for software architects; training, soft skills, coaching, mentoring, education, and certification; etc.

Mar 31-Apr 01 9th **International Workshop on Formal Approaches for Advanced Computing Systems** (FAACS'2025). Topics include: integration between formal methods and software architecture, architecture description languages and metamodels, model-driven engineering, approaches and tools for verification and validation, reports on practical experience in the application of formal methods to industrial case studies, etc.

Mar 31 - Apr 04 18th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2025), Naples, Italy. Topics include: formal verification; replications, empirical studies, case studies, experience reports; software reliability; static and dynamic analysis; test automation; testability, test design, and adequacy criteria; testing and development processes; testing, debugging, and repair tools; testing in specific domains (embedded/cyber-physical systems, concurrent, distributed, real-time systems, ...); testing of non-functional properties such as security; etc. Deadline for submissions: January 3, 2025 (workshop papers).

April 01-03 **International Conference on Advances in Parallel and Distributed Computing** (APDC'2025), Downtown Oklahoma City, OK, USA & Online. Topics include: parallel computing; parallel/distributed applications; programming models/benchmarks/tools; formal methods and theoretical foundations; parallel, distributed, and concurrent systems; etc. Deadline for submissions: January 19, 2025 (abstracts, papers).

April 07-08 11th **International Conference on Fundamentals of Software Engineering** (FSEN'2025), Västeräs, Sweden. Topics include: all aspects of formal methods, especially those related to advancing the application of formal methods in the software industry and promoting their integration with practical engineering techniques; models of programs and software systems; software specification, validation, and verification; software testing; software architectures and their description languages; integration of formal and informal methods; component-based and service-oriented software systems; cyber-physical software systems; model checking and theorem proving; software verification; CASE tools and tool integration; industrial applications; etc.

April 07-10 31st **International Working Conference on Requirements Engineering: Foundation for Software Quality** (REFSQ'2025), Barcelona, Spain. Theme: "Social REsponsibility". Deadline for submissions: February 7, 2025 (papers to workshops, education and training, posters and tools, and doctoral symposium).

April 08-11 20th **European Dependable Computing Conference** (EDCC'2025), Lisbon, Portugal. Topics include: latest ideas and results on theory, experiments, techniques, systems and tools for the design, validation, operation and evaluation of dependable and secure computing systems; hardware and software architecture of dependable systems; dependability and security modelling, evaluation, and tools; safety-critical systems design and analysis; mixed-criticality systems design and evaluation; testing and validation methods; dependability and security of: artificial intelligence systems, cyber-physical systems, e.g. intelligent vehicles, (industrial) Internet of Things, ...; etc.

Apr 26 – May 04 47th **International Conference on Software Engineering** (ICSE'2025), Ottawa, Ontario, Canada. Topics include: the full spectrum of Software Engineering (SE), trustworthy AI for SE; AI-assisted software design and model driven engineering; mining software repositories; software metrics (and measurements); software design methodologies, principles, and strategies; architecture quality attributes, such as security, privacy, performance, reliability; modularity and reusability; dependency and complexity analysis; patterns and anti-patterns; technical debt in design and architecture; formal methods and model checking; reliability, availability, and safety; resilience and antifragility; design for dependability and security; vulnerability detection to enhance software security; dependability and security for embedded and cyber-physical systems; evolution and maintenance; API design and evolution; software reuse; refactoring and program differencing; program comprehension; reverse engineering; environments and software development tools; human and social aspects (focusing on programming languages, environments, and tools supporting individuals, teams, communities, and companies; focusing on software development processes; ...); modeling and model-driven engineering; variability and product lines; modeling languages, techniques, and tools; empirical studies on the application of model-based engineering; software testing; automated test generation techniques such as fuzzing, search-based approaches, and symbolic execution; testing and analysis of non-functional properties; program analysis; debugging and fault localization; runtime analysis and/or error recovery;

April 27-28 13th **International Conference on Formal Methods in Software Engineering** (FormaliSE'2025). Topics include: approaches, methods and tools for verification and

validation; formal approaches to safety and security related issues; scalability of formal method applications; integration of formal methods within the software development lifecycle; model-based engineering approaches; correctness-by-construction approaches for software and systems engineering; application of formal methods to specific domains, e.g., autonomous, cyber-physical, intelligent, and IoT systems; formal methods in a certification context; case studies developed/analyzed with formal approaches; experience reports on the application of formal methods to real-world problems; guidelines to use formal methods in practice; usability of formal methods; etc.

Apr 27 - May 03 37th **International Conference on Software Engineering Education and Training** (CSEET'2025), Ottawa, Ontario, Canada. Topics include: all dimensions of learning and teaching in the area of software engineering.

Apr 27 – May 03 47th **International Conference on Software Engineering** (ICSE'2025), Ottawa, Ontario, Canada.

May 03-08 28th ETAPS **International Joint Conferences on Theory and Practice of Software** (ETAPS'2025), Hamilton, Canada. Deadline for submissions: January 9, 2025 (FASE, FoSSaCS artifacts), January 15, 2025 (nominations ETAPS Doctoral Dissertation Award).

☺ May 04 16th **Workshop on Programming Language Approaches to Concurrency- and communication-cEntric Software** (PLACES'2025). Topics include: general area of programming language approaches to concurrency, communication and distribution, ranging from foundational issues, through language implementations, to applications and case studies; design and implementation of programming languages with first class concurrency and communication primitives; models for concurrent and distributed systems; concurrent data types, objects and actors; verification and program analysis methods for safe and secure concurrent and distributed software; etc. Deadline for submissions: February 20, 2025 (abstracts), February 28, 2025 (papers).

May 04 1st **International Workshop on Verification of Scientific Software** (VSS'2025). Topics include: ways to specify scientific software; effective verification techniques for programs that use concurrency interfaces in scientific computing; precise reasoning about floating-point computations; case studies applying verification tools to scientific software; methods to decompose verification problems for scientific programs, such as function contracts; etc. Deadline for submissions: February 1, 2025.

May 05-08 24rd **European Symposium on Programming** (ESOP'2025). Topics include: fundamental issues in the specification, design, analysis, and implementation of programming languages and systems, such as programming paradigms and styles, methods and tools to specify and reason about programs and languages, programming language foundations, methods and tools for implementation, concurrency and distribution, etc.

May 07-08 31st **International Symposium on Model Checking of Software** (SPIN'2025). Topics include: automated tool-based techniques for the analysis of software as well as models of software, for the purpose of verification and validation. Deadline for submissions: February 13, 2025, February 27, 2025 (tool artifacts), March 12, 2025 (non-tool artifacts).

May 06-09 18th **Cyber-Physical Systems and Internet of Things Week** (CPS-IoT Week'2025), Irvine, USA. Event includes: 5 top conferences, HSCC, ICCPS, IoTDI, IPSN, and RTAS, as well as poster and demo sessions, workshops, tutorials, competitions, industrial exhibitions, PhD forums, and summits.

☺ May 06-09 31st IEEE **Real-Time and Embedded Technology and Applications Symposium** (RTAS'2025). Topics include: applications with timing requirements; real-time and embedded operating systems; application profiling, WCET analysis, compilers, tools, benchmarks and case studies; modelling languages, modelling methods, model learning, model validation and calibration; scheduling and resource allocation; schedulability and response time analyses; verification and validation methodologies; etc.

May 06-09 16th ACM/IEEE **International Conference on Cyber-Physical Systems** (ICCPS'2025). Topics include: safety and resilience for CPS; software platforms and systems for CPS; specification languages and requirements; design, optimization, and synthesis; testing, verification, certification, assurance; security, trust, and privacy in CPS; tools, testbeds,

demonstrations and deployments; CPS applications in power systems, infrastructure networks, transportation, healthcare, automotive, aerospace, etc. Deadline for submissions: February 7, 2025 (posters, demos).

May 12-14 — 25th annual **High Confidence Software and Systems Conference** (HCSS'2025), Annapolis, Maryland, USA. Topics include: use of advanced languages and tools, etc. Deadline for submissions: January 13, 2025 (abstracts).

May 12-16 — 28th **Ibero-American Conference on Software Engineering** (CIbSE'2025), Ciudad Real, Spain. Topics include: community-based software engineering (SE) (e.g., open source, crowdsourcing); ethics in SE; industrial experience reports in SE; software architecture and variability; software ecosystems and systems of systems; SE education and training; SE for emerging application domains (cyber-physical systems, Internet of Things, ...); software evolution and modernisation; software modeling and model-driven engineering; software processes; software product lines and processes; software quality, quality models and technical debt management; software reliability; software repository mining and software analytics; software reuse; software testing; etc. Deadline for submissions: January 17, 2025 (abstracts), January 31, 2025 (papers, doctoral symposium, journal first).

May 20-22 — 17th **Software Quality Days** (SWQD'2025), Munich, Germany. Theme: "Balancing Software Innovation and Regulatory Compliance" Topics include: all topics related to software and systems quality; methods and tools for constructive and analytical quality assurance; testing of software and software-intensive systems; process improvement for development and testing; automation in quality assurance and testing; domain-specific quality issues such as embedded, medical, and automotive systems; continuous integration, deployment, and delivery; project and risk management; secure coding, software engineering, and system design; detection and prevention of vulnerabilities and security threats; etc.

☺ May 26-28 — 28th IEEE **International Symposium On Object/Component/Service-Oriented Real-Time Distributed Computing** (ISORC'2025), Toulouse, France. Topics include: all aspects of object real-time distributed computing (ORC) technology; such as software architectures for distributed and/or real-time computing; cybersecurity, and trust for distributed and/or real-time IoT systems; formal verification and model checking for distributed and real-time computing; dependability, fault tolerance, and resilience; distributed and/or real-time computing applications in IoT, CPS, edge-cloud, etc. Deadline for submissions: January 8, 2025 (papers), January 15, 2025 (workshops).

May 26-31 — 6th **Programming Language Implementation Summer School** (PLISS'2025), Bertinoro, Italy. Topics include: current research and future trends in programming language design and implementation, such as compiler and allocator optimisations, human and design factors in language design and implementation, language security and interactions with hardware, etc.

June 03-07 — 39th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2025), Milan, Italy. Topics include: research in high performance computing in parallel and distributed processing; real-world applications that use parallel and distributed computing concepts; experiments and performance-oriented studies in the practice of parallel and distributed computing; programming models, compilers, and runtime systems (ranging from the design of parallel programming models and paradigms, to languages and compilers supporting these models and paradigms, to runtime and middleware solutions); etc. Deadline for submissions: January 17 - February 14, 2025 (workshop papers).

♦ June 10-13 — 29th **Ada-Europe International Conference on Reliable Software Technologies** (AEiC'2025), Paris, France. Organized by Ada-Europe and Ada-France. Deadline for submissions: February 7, 2025 (journal track papers), February 24, 2025 (industrial track and work-in-progress papers, tutorial and workshop proposals). Submit early: acceptance decisions for the journal track and workshops are made on a rolling basis! #AEiC2025 #AdaEurope #AdaProgramming

June 12-13 — **Software Technologies: Applications and Foundations** (STAF'2025), Koblenz, Germany. Topics include: practical and foundational advances in software technology. Deadline for early registration: May 9, 2025.

June 12-13 — 18th ACM SIGPLAN **International Conference on Software Language Engineering** (SLE'2025). Topics include: software language engineering in general, rather than engineering a specific software language, such as software language design and implementation, software language validation (verification and formal methods for languages, testing techniques for languages, simulation techniques for languages, ...), software language integration and composition, software language maintenance

(software language reuse; language evolution; language families and variability, language and software product lines), domain-specific approaches for any aspects of SLE (design, implementation, validation, maintenance), empirical evaluation and experience reports of language engineering tools (user studies evaluating usability, performance benchmarks, industrial applications), etc. Deadline for submissions: February 7, 2025 (abstracts), February 14, 2025 (papers).

June 16-20 — 20th **International Federated Conference on Distributed Computing Techniques** (DisCoTec'2025), Lille, France. Includes the COORDINATION, DAIS, and FORTE conferences. Topics include: a broad spectrum of distributed computing subjects, from theoretical foundations and formal description techniques, testing and verification methods, to language design and system implementation approaches. Deadline for submissions: February 7, 2025 (papers), February 10, 2025 (workshops, tutorials), mid April, 2025 (workshop papers).

June 23-27 — 33rd ACM **International Conference on the Foundations of Software Engineering** (FSE'2025), Trondheim, Norway. Topics include: debugging and fault localization; dependability, safety, and reliability; embedded software, safety-critical systems, and cyber-physical systems; model checking; model-driven engineering; parallel, distributed, and concurrent systems; program analysis; programming languages; software architectures; software engineering education; software evolution; software security; software testing; software traceability; symbolic execution; tools and environments; etc. Deadline for submissions: February 25, 2025 (workshop papers).

June 25-28 — 34th ACM SIGSOFT **International Symposium on Software Testing and Analysis** (ISSTA'2025), Trondheim, Norway. Topics include: theory, design, implementation, optimization, testing, and analysis of software systems, programs and programming languages. Deadline for submissions: January 3, 2025 (impact paper award nominations), January 10, 2025 (workshops).

☺ June 30 – July 04 — 39th **European Conference on Object-Oriented Programming** (ECOOP'2025), Bergen, Norway. Topics include: all topics related to programming languages, software development, systems and applications; all practical and theoretical investigations of programming languages, systems and environments; innovative solutions to real problems as well as evaluations of existing solutions. Deadline for submissions: January 6, 2025 (round 1), January 24, 2025 (workshops), January 31, 2025 (nominations for Dahl-Nygaard prizes), March 5, 2025 (round 2).

July 02-04 — 28th **International Conference on Engineering of Complex Computer Systems** (ICECCS'2025), Hangzhou, China. Topics include: engineering of complex computer systems, such as distributed systems, autonomous intelligent systems, and cyber-physical systems; requirements, modeling and formal methods; software engineering; simulation, testing, and validation; security, reliability and dependability; etc. Deadline for submissions: January 28, 2025 (abstracts), February 4, 2025 (full papers).

July 08-11 — 37th **Euromicro Conference on Real-Time Systems** (ECRTS'2025), Brussels, Belgium. Topics include: all aspects of timing requirements in computer systems; elements of time-sensitive software systems, such as operating systems, hypervisors, middlewares and frameworks, programming languages and compilers, runtime environments, ...; real-time applications, such as modeling, design, simulation, testing, debugging, and evaluation in domains such as automotive, avionics, control systems, industrial automation, robotics, space, railways telecommunications, ...; foundational scheduling and predictability questions, such as schedulability analysis, synchronization protocols, multi-core scheduling, ...; static and dynamic techniques for resource demand estimation, such as classic worst-case execution time (WCET) analysis, ...; formal methods for the verification and validation of real-time systems, such as model checking, computer-assisted proofs, ...; the interplay of timing predictability and other non-functional qualities, such as reliability, quality of control, testability, scalability, ...; etc. Deadline for submissions: February 28, 2025 (papers).

August 25-30 — 36th **International Conference on Concurrency Theory** (CONCUR'2025), Aarhus, Denmark. Co-located with QEST+FORMATS and FMICS as part of CONFEST 2025 Topics include: theory and practice of concurrent systems; verification and analysis techniques for concurrent systems (abstract interpretation, model checking, race detection, run-time verification, static analysis, testing, theorem proving, type systems, security analysis, ...); distributed/parallel algorithms and concurrent data structures (design, analysis, complexity, correctness, fault tolerance, reliability, availability, consistency, ...); theoretical foundations, tools, and empirical evaluations of architectures, execution environments, and software development for concurrent systems such as multiprocessor and multi-core architectures;

compilers and tools for concurrent programming; programming models such as component-based, object-oriented, ...; etc. Deadline for submissions: April 1, 2025 (abstracts), April 7, 2025 (papers).

September 09-12 44th **International Conference on Computer Safety, Reliability and Security** (SafeComp'2025), Stockholm, Sweden. Topics include: all aspects related to the development, assessment, operation, and maintenance of safety-related and safety-critical computer systems; fault detection and recovery mechanisms; safety guidelines and standards; safety/security co-engineering and trade-offs; safety and security qualification, quantification, assurance and certification; threats and vulnerability analysis; model-based analysis, design, and assessment; formal methods for verification, validation, and fault tolerance; testing, verification, and validation methodologies and tools; 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, smart grids, SCADA; medical devices and healthcare; surveillance, defensed, emergency & rescue; logistics, industrial automation, off-shore technology; education & training. Deadline for submissions: 7 February 2025 (workshops, abstracts), 14 February 2025 (full papers).

September 10-12 51st **Euromicro Conference on Software Engineering and Advanced Applications** (SEAA'2025), Salerno, Italy. Topics include: information technology for software-intensive systems; tracks on Cyber-Physical Systems (CPS), Emerging Computing Technologies (ECT), Model-Driven Engineering and Modeling Languages (MDEML), Software Process and Product Improvement (SPPI), Practical Aspects of Software Engineering (KKIO), etc. Deadline for paper submissions: April 15, 2025.

☺ October 12-18 ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2025), Singapore.

☺ Oct 12-25 **Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2025). Deadline for submissions: October 15, 2024 (round 1), March 25, 2025 (round 2).

☺ Nov 05-07 33rd **International Conference on Real-Time Networks and Systems** (RTNS'2025), Pisa, Italy. Topics include: real-time applications design and evaluation (automotive, avionics, space, railways, telecommunications, process control, ...), 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, security, ...), software technologies for real-time systems (model-driven engineering, programming languages, compilers, WCET-aware compilation and parallelization strategies, middleware, Real-Time Operating Systems, ...), formal specification and verification, real-time distributed systems, etc. Deadline for submissions: January 30, 2025 (1st round), May 29, 2025 (2nd round), August 14, 2025 (3rd round).

☺ Dec 02-05 46th IEEE **Real-Time Systems Symposium** (RTSS'2025), Boston, Massachusetts, USA Deadline for submissions: October 15, 2023 (hosting proposals), May 22, 2025 (papers).

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

Call for Participation
12th Ada Developer Room at FOSDEM 2025
Sunday 2 February 2025, Brussels, Belgium

Organized in cooperation with
Ada-Belgium[1] and Ada-Europe[2]

FOSDEM[3], 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 2025 edition takes place on Saturday 1 and Sunday 2 February. It is free to attend and no registration is necessary.

In this edition, the Ada FOSDEM community organizes once more a set of presentations related to Ada and Free or Open Software in a s.c. Developer Room. The "Ada DevRoom" at FOSDEM 2025 is held on the morning of the 2nd day, and offers a variety pf presentations on the Ada programming language, tools and projects: a total of 11 Ada-related presentations by 11 authors from 7 countries!

Program overview:

- *Welcome to the Ada DevRoom,* by Fernando Oleo Blanco, Spain, and Dirk Craeynest, Belgium
- *Updates on the Ada Ecosystem,* by Fernando Oleo Blanco, Spain
- *Get started with Ada in 2 minutes or less!,* by A.J., USA
- *Advent of Compression: writing a working BZip2 encoder in Ada from scratch in a few days,* by Gautier de Montmollin, Switzerland
- *Ada and Mini-Ada: a solution to the two-language problem,* by Gautier de Montmollin, Switzerland
- *Understanding liquid types, contracts and formal verification with Ada/SPARK,* by Fernando Oleo Blanco, Spain
- *The state of Rust trying to catch up with Ada,* by Oli Scherer, Germany
- *Cryptography in SPARK: building the foundation with constant-time bigints,* by César Sagaert and Fabien Chouteau, France
- *Multiword Arithmetic and Parallel Computing,* by Jan Verschelde, USA
- *Developing device drivers for Ironclad using Ada,* by streaksu
- *AdaBots - programmable minetest bots,* by Tama McGlinn and Rudolf Batke, the Netherlands

The Ada at FOSDEM 2025 web-page has all details, such as the full schedule, abstracts of presentations, biographies of speakers, and pointers to more info, including live video streaming and recordings afterwards. For the latest information at any time, contact Fernando Oleo Blanco <irvise@irvise.xyz>, or see:

https://fosdem.org/2025/schedule/track/ada/

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/25/250202-fosdem.html

1 http://www.cs.kuleuven.be/~dirk/ada-belgium/
2 http://www.ada-europe.org/
3 https://fosdem.org/2025/

29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025)
10-13 June 2025, Paris, France

Conference Chair
Jean-Pierre Rosen
rosen@adalog.fr
Adalog & Ada-France

Journal track Co-chairs
Laurent Pautet
laurent.pautet@telecom-paris.fr
Telecom Paris
Sara Royuela
sara.royuela@bsc.es
Barcelona Supercomputing Center

Industrial track Co-chairs
Daniela Cancila
daniela.cancila@cea.fr
CEA LIST
Laurent Gouzenes
lgouzenes@pactenovation.fr
Pacte-Novation

Work-in-progress track Co-chairs
Hai Nam Tran
hai-nam.tran@univ-brest.fr
University of Brest
Anish Bhobe
anish.bhobe@telecom-paris.fr
Telecom Paris

Workshop Chair
Anish Bhobe
anish.bhobe@telecom-paris.fr
Telecom Paris

Tutorial Chair
Robert Cholay
robert.cholay@systerel.fr
Systerel

Exhibition & Sponsorship Chair
Ahlan Marriott
ahlan@Ada-Switzerland.ch
White Elephant GmbH

Finance Chair
Paul Duquennoy
paul.duquennoy@free.fr

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

Webmaster
Hai Nam Tran
hai-nam.tran@univ-brest.fr
University of Brest

Local Chair
Pierre Jouvelot
pierre.jouvelot@minesparis.psl.eu
Mines Paris, PSL University

General Information

The **29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025)** will take place in Paris, France. The conference schedule comprises a journal track, an industrial track, a work-in-progress track, a vendor exhibition, parallel tutorials, and satellite workshops.

- Journal track papers present research advances supported by solid theoretical foundation and thorough evaluation.
- Industrial track contributions highlight the practitioners' side of a challenging case study or industrial project.
- Work-in-progress track papers illustrate novel research ideas that are still at an initial stage, between conception and first prototype.
- Tutorials guide attenders through a hands-on familiarization with innovative developments or with useful features related to reliable software.
- Workshops provide discussion forums on themes related to the conference topics.

Schedule

7 February 2025	Deadline for submission of journal track papers (extended)
24 February 2025	Deadline for submission of industrial track papers, work-in-progress papers, and tutorial and workshop proposals
28 March 2025	First round notification for journal track papers, and notification of acceptance for all other types of submissions
10-13 June 2025	Conference

Scope and 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, 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:

- Formal and model-based engineering of critical systems
- High-integrity systems and reliability
- AI for high-integrity systems engineering
- Real-time systems
- Ada language
- Applications in relevant domains

More specific topics are described on the conference web page.

Call for Journal Track Submissions

Following a journal-first model, this edition of the conference includes a journal track, which seeks original and high-quality papers that describe mature research work on the conference topics. Accepted journal track papers will be published in a Special Issue of Elsevier JSA – the *Journal of Systems Architecture* (Q1 ranked, CiteScore 8.5, impact factor 3.7). Accordingly, the conference is listed as "Journal Published" in the latest update of the CORE Conference Ranking released in August 2023. Contributions must be submitted by **7 February 2025 (extended)**. Submissions should be made online at https://www.editorialmanager.com/jsa/, selecting the "VSI:AEiC2025" option as article type of the paper. General information for submitting to the JSA can be found at the Journal of Systems Architecture website.

JSA has adopted the Virtual Special Issue model to speed up the publication process, where Special Issue (SI) papers are published in regular issues, but marked as SI papers. Acceptance decisions are made on a rolling basis. Therefore, authors are encouraged to submit papers early, and need not wait until the submission deadline. Authors who have successfully passed the first round of review will be invited to present their work at the conference. Ada-Europe will waive the Open Access fees for the first four accepted papers (whose authors do not already enjoy Open Access agreements). Subsequent papers will follow JSA regular publishing track.

Prospective authors may direct all enquiries regarding this track to the corresponding chairs, Laurent Pautet and Sara Royuela.

Call for Industrial Track Submissions

The conference seeks industrial-practitioner presentations that deliver insight on the challenges of developing reliable software. Especially welcome kinds of submissions are listed on the conference web site. Given their applied nature, such contributions will be subject to a dedicated practitioner-peer-review process. Interested authors shall submit a one-to-two pages abstract, by **24 February 2025**, via EasyChair at https://easychair.org/conferences/?conf=aeic2025, selecting the "Industrial Track". The format for submission is strictly in PDF, following the Ada User Journal style. Templates are available at http://www.ada-europe.org/auj/guide.

The abstracts of the accepted contributions will be included in the conference booklet. The corresponding authors will get a presentation slot in the prime-time technical program of the conference and will also be invited to expand their contributions into full-fledged articles for publication in the *Ada User Journal*, which will form the proceedings of the industrial track of the Conference. Prospective authors may direct all enquiries regarding this track to its chairs, Daniela Cancila and Laurent Gouzenes.

Call for Work-in-progress Track Submissions

The work-in-progress (WiP) track seeks two kinds of submissions: (a) ongoing research and (b) early-stage ideas. Ongoing research submissions are 4-page papers describing research results that are not mature enough to be submitted to the journal track. Early-stage ideas are 1-page papers that pitch new research directions that fall within the scope of the conference. Both kinds of submissions must be original and shall undergo anonymous peer review. Submissions by recent MSc graduates and PhD students are especially sought. Authors shall submit their work by **24 February 2025**, via EasyChair at https://easychair.org/conferences/?conf=aeic2025, selecting the "Work-in-progress Track". The format for submission is strictly in PDF, following the Ada User Journal style. Templates are available at http://www.ada-europe.org/auj/guide.

The abstract of the accepted contributions will be included in the conference booklet. The corresponding authors will get a presentation slot in the prime-time technical program of the conference and will also be offered the opportunity to expand their contributions into 4-page articles for publication in the *Ada User Journal*, which will form the proceedings of the WiP track of the conference. Prospective authors may direct all enquiries regarding this track to the corresponding chairs, Hai Nam Tran and Anish Bhobe.

Call for Tutorials

The conference seeks tutorials in the form of educational seminars on themes falling within the conference scope, with an academic-for-practitioner slant, including hands-on or practical elements. 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), the intended level of the contents (introductory, intermediate, or advanced), and a statement motivating attendance. Tutorial proposals shall be submitted at any time but no later than **24 February 2025** to the respective chair, Robert Cholay, by email, with subject line: "[AEiC 2025: tutorial proposal]". The authors of accepted full-day tutorials will receive a complimentary conference registration, halved for half-day tutorials. The *Ada User Journal* will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

The conference welcomes satellite workshops centred on themes that fall within the conference scope. Proposals may be submitted for half- or full-day events, to be scheduled at either end of the AEiC conference. Workshop organizers shall also commit to producing the proceedings of the event, for publication in the *Ada User Journal*. Workshop proposals shall be submitted at any time but no later than **24 February 2025** to the respective chair, Anish Bhobe, by email, with subject line: "[AEiC 2025: workshop proposal]". Once submitted, each workshop proposal will be evaluated by the conference organizers as soon as possible.

Call for Exhibitors

The conference will include a vendor and technology exhibition. Interested providers should direct inquiries to the Exhibition & Sponsorship Chair, Ahlan Marriott.

Venue

The conference will take place at Mines Paris. Mines Paris - PSL, a founding member of Université PSL, is a leading French engineering school and the French leader institution in research partnerships. Founded 240 years ago to help spur the energy efforts called by the Industrial Revolution, it has been since training engineers in a wide spectrum of scientific disciplines. With about 1,500 students, including 100 PhD graduates per year, Mines Paris - PSL hosts 18 research centres and 5 academic departments. It is located along the Luxembourg gardens, next to the Quartier Latin, and is close to public transportation, including line B of the RER to Charles De Gaulle airport (CDG).

Paris, the capital city of France, is renowned for its rich history, stunning architecture, and vibrant culture. Often referred to as "The City of Light," Paris is home to iconic landmarks such as the Eiffel Tower, the Louvre Museum, and Notre-Dame Cathedral. The city is a global centre for art, fashion, gastronomy, science and culture, attracting millions of visitors each year.

Front picture generated with the assistance of AI (DALL·E 2). Back picture provided by Mines Paris.

It Is Time to Care for Ada!

T. Vardanega
Ada User Society, President, email: president@ada-user.org

1 A rather strange introduction

I confess, I am a voracious book reader, and being (luckily) versed in quite a few tongues, I enjoy reading books frequently (but not always) without needing translation. In fact, I find languages fascinating, for all they say about people's culture, history, and traits.

As a student in Computer Science in the late 1980s, I was imbued with Noam Chomsky's linguistic "generative grammar" theory. That theory sees specific languages as second-order derivatives of a single universal grammar, innate in the human mind. That theory, with its hierarchy of formal grammars, has laid the foundation for the theory and practice of language compilers. The principal tenet of Chomsky's theory is that grammar precedes language.

Interestingly, since then, linguistics has pushed the study of the relation between grammar and language much further. A splendid book by Swedish author Sverker Johannsson, "The dawn of language", which I thoroughly enjoyed (though not in Swedish, regrettably :-) and which I warmly recommend, posits that all languages originate from social requirements, and precede grammar. Grammars, linguists now maintain, become necessary when the social needs become more sophisticated and indirect to the point of requiring multiple levels of subordinates. The book author supports this claim with a rich and fascinating weaving of archaeological, anthropological, neurological and linguistic findings, which reflect how modern science has become multidisciplinary.

This is the point I wanted to come to in this perhaps unusual introduction: "languages originate from social needs". Social needs form in the presence of stable social gatherings. Some by necessity; others by election.

Then perhaps it can also be said that programming languages reflect some specific sort of "social needs"; thus, at some level, also the "ideology", a view of the world, of how a program tells what is (wanted) to happen and, by virtue of its compiler, hence its grammar, how it abides to that intent at run time.

In my view, this tenet is especially (though not exclusively) true of Ada. It is therefore consequent that Ada can be seen as the expression of the social needs – within the precincts of programming – of its constituency.

2 Language users and responsibilities

Social needs are intrinsically and eminently communitarian. Languages that reflect social needs cause users to hold shared responsibility for how their language should evolve and how it should represent their take in it.

This should also be true of programming languages, arguably. And, I think, it should be true also of Ada.

In the social gathering of a programming language there are diverse roles and different levels of responsibility, but there also is one single common allegiance. There are language designers (who turn user needs into language features), language lawyers (who turn design features into language specifications), language implementors (who turn specifications into usable technology), language users, language educators, language evangelists, language historians. From different angles, they all factually form one and the same community. Of course, a critical value chain extends across some of the above roles, and its economic value allows part of the work required for the language to continue to exist, to be rewarded monetarily and to produce profit. The monetary element of the value chain, however, should never reduce the ecology of a language community to the bilateral relation between customers and suppliers. Arguably, there is more to a programming language than paying customers and technology providers, and it is so because of the very nature of language.

I am old enough to have observed and learned that there are moments in the life of programming languages where very little is required of some members of the language community, as most if not all seems to happen via paid roles and tasks. There are other times, however, in which the economic sustainability of those paid roles and tasks is threatened. It is at those times especially that the entire community, regardless of the assigned roles in it, is urged to become more directly responsible for the continued existence of the language. This is one such moment, I think, for Ada, and this understanding has prompted me into action.

3 A call to action

You may perhaps know that I have been president of Ada-Europe for 20 years, until September 2024. The charter of that association, what it stands for, is very well described in the opening line of the "What is Ada-Europe?" section of its web site. Ada-Europe, that text reads, aims to spread the use and the knowledge of Ada and to promote its introduction into academic and research establishments. As its namesake suggests, it does so from the perspective of Europe, understood in terms of geographical reach and scope more than anything else.

Over time, however, from my vantage point that ranged from chairing the board of Ada-Europe to seating in WG 9 (the body under ISO that oversees to the maintenance of the language standard), and being member of the ARG (the expert group tasked by WG 9 to carry out the technical work involved in evolving the language specification), I have

become increasingly aware that something more direct and wider than representing a single geographical continent is required of an international organization intended to keep Ada alive.

My first encounter with Ada was in the early stages of the process denominated “Ada 9X”, which resulted in the Ada 95 version of the language specification. That was an important moment of transition from MIL-STD – you look this term up yourself if you don’t know what that means – to ISO. Being sanctioned under a civil international standard signified that Ada wanted to be a global resource inscribed within a worldwide communitarian process. Conformance with ISO procedures carries – and has carried – considerable costs, especially service time and other overheads of WG 9 and ARG.

Over time, the sustenance of those costs had been progressively shouldered by a single commercial vendor, with very little direct support from any other entity, including Ada associations. When a single, private, entity comes to bear all the costs of keeping a social benefit alive, then the question of ownership, hence its real “social” nature, and of the associated decisional authority naturally arises, and the prospect of “privatizing” becomes a natural option. If that were to happen, then Ada would cease to be the language of a “social gathering”. In other words, the time had come for the global Ada community, assuming that one still existed, to take direct and collective action for the continuation of Ada as an international standard, under a communitarian process, sharing as much of the load and responsibility as possible.

This was the thinking behind the creation of the Ada User Society, with the single and specific task to create the place for all friends of Ada to gather around it, provide direct and indirect support to the effort of keeping Ada under ISO, not so much for ISO’s sake, but for what that means symbolically and practically. That objective requires: (1) membership to the Ada User Society from anybody anywhere, who feels that Ada continues to have value in the programming language landscape; (2) voluntary contribution to the technical work entailed by the maintenance of the language specifications, also including reference implementations; (3) financial donations to help sustain the costs that necessarily arise in keeping conformance with the ISO process.

This was the reason for creating the Ada User Society, which was founded on 11 July 2024, almost one full year ago, although you may have seen very little of it to date. The charter of the Ada User Society is very specific, perhaps even narrow, but it is altogether essential to there being an Ada language to speak of at all.

This short piece, graciously featured by the Ada User Journal, the periodical of Ada-Europe (how interesting that there is such an evident naming match 😊) warmly invites your membership and your support to that collective and communitarian goal, which I hope and trust you subscribe to. Hence, please, go visit https://ada-user.org/ and join the Ada User Society!

Implementing Unsafe Features on top of a Safe Virtual Machine

S. Tucker Taft
AdaCore, 130 W 150th St., New York, NY, 10001; email: taft@adacore.com

Abstract

It is common to translate a programming language to an intermediate representation as part of interpreting or compiling a program written in the language. The question is what are the implications when we try to implement normally unsafe features, such as pointers into a heap with user-controlled deallocation, on top of a virtual machine based on an intermediate language that is inherently safe and provides no user-controlled deallocation. A similar question arises in the implementation of exceptions on top of a virtual machine that has no direct support for exceptions but has lightweight threading. This paper will describe how we are addressing these two challenges as part of building an Ada implementation on top of a parallel virtual machine, and evaluate the result.

Keywords: Ada, unsafe features, safe intermediate language, exception handling, memory safety, parallel virtual machine.

1 Ada on top of a Parallel Virtual Machine

To support the prototyping of features proposed for Ada 2022 [1] and future versions of the Ada standard, we have been implementing parts of Ada on top of a parallel virtual machine developed for the ParaSail parallel programming language [2]. The ParaSail virtual machine has built-in support for lightweight parallelism coupled with a safe, data-race-free, ownership-based storage model. Both an interpreter and a compiler to the LLVM intermediate language exist for the ParaSail virtual machine. Two features not supported by the ParaSail virtual machine are user-level pointers, and exceptions. This paper will describe our efforts to implement the corresponding Ada features (access types and exception handling) on top of the ParaSail virtual machine.

2 Supporting pointers without pointers

The fundamental property of a pointer is that it identifies an object whose lifetime might outlive the scope where the object was created, with the identifier being something that can be stored in multiple other objects. In a virtual machine that provides only ownership-based storage, there is normally no way to create multiple copies of a reference to an object, if at least one of the references allows the object to be updated. In general, the exclusive write property of object references is the definition of an ownership-based storage model [2].

In a language that does not have pointers, one alternative is to use an array, with indices into the array acting as pointers. However, to make this work with the same convenience provided by more traditional pointers, these "arrays" need to have additional properties. They need to be able to grow dynamically as the number of pointed-to objects grows, and they need to be able to hold elements of varying sizes, rather than requiring that all elements of the array are of uniform size. It turns out that the ParaSail virtual machine provides the building blocks needed to implement such an array. Every type of object in ParaSail has one additional "null" value that takes no space. A component or variable that is declared "optional" is permitted to hold such a null value. The ParaSail virtual machine automatically handles the reclamation of the storage used for a non-null value of a variable or component, if it is set to null. This basic capability makes it straightforward to create extensible and shrinkable objects, and an extensible array of extensible objects is readily constructed.

Given that the ParaSail virtual machine provides these building blocks, in this section we will examine how they are best used to support the various kinds of access types provided by Ada. We will also examine how Ada's so-called "unchecked" deallocation works depending on the model chosen, and how that affects what is considered one of the "unsafe" features of Ada.

2.1 Aliased Objects and Access Types

Aliased objects in Ada are objects that may be the target of a pointer, which in Ada are called *access values*, values of an *access type*. Ada has several kinds of access types:

- Pool-specific access types: values of such a type may designate only (heap-resident) aliased objects created within a particular *storage pool* (a local heap), created using an *allocator* (of the form **new** T'(...)).
- General access-to-variable types: values of such a type may designate aliased variable objects created using an allocator, or stack-resident aliased variable objects.
- Access-to-constant types: values of such a type may designate aliased objects, constant or variable, created using an allocator or resident on the stack.

Pool-specific access types are the most straightforward to support in the ParaSail virtual machine (PSVM). When such a type is declared, an extensible vector of the designated type may be created to represent the storage pool for this type, and as objects are created using allocators, an element is

added to the end of the vector, and the index within the vector may be used to represent the access value of the type.

General access types are somewhat more difficult to support on the PSVM, in that values of such a type may designate objects in the storage pool associated with the type, but they may also designate objects in other storage pools, and on the stack. This implies that a simple index into a storage pool "vector" will not be adequate to represent such values. At a minimum we will need some sort of identifier of the storage pool, plus the index into the storage pool. But this is not enough to handle stack-resident aliased objects. Such aliased objects do not reside in a storage pool, at least at the Ada language level. Handling such objects will generally require the creation of an implicit storage pool, and then the creation of an object within that storage pool, replacing the stack-resident object with a pointer to this storage-pool-resident object, with an implicit dereference provided on any normal reference to such a stack-resident aliased object. Similar considerations apply to aliased components of other objects. They will similarly need to be moved into a storage pool and replaced with an implicitly-dereferenced access value.

Access-to-constant types do not provide any additional challenges, since they are merely general access types that are allowed to designate aliased objects that are either variables or constants. Whatever solution works for a general access type should work for an access-to-constant type.

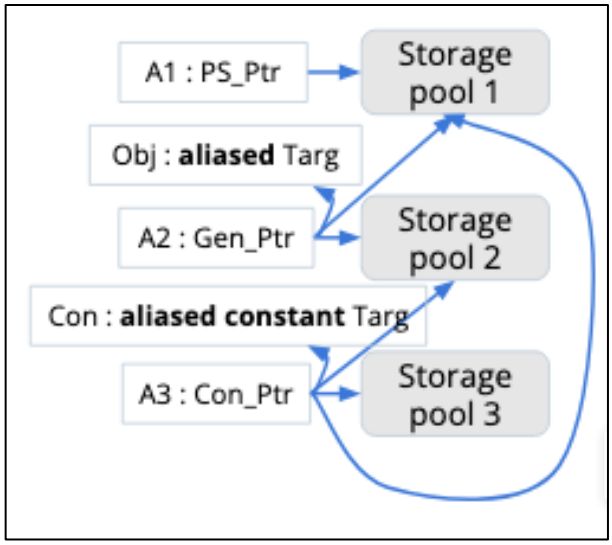


Figure 1 Kinds of Access Types

In Figure 1, we illustrate a pool-specific access type (PS_Ptr), a general access type (Gen_Ptr), and an access-to-constant access type (Con_Ptr). Values of type PS_Ptr may only point into its associated storage pool, Storage pool 1. Values of type Gen_Ptr may point into its own storage pool (Storage pool 2), at a stack-resident aliased object such as Obj, as well as into any longer-lived storage pool (such as Storage pool 1). Values of type Con_Ptr may point into its own storage pool (Storage Pool 3), as well as at longer lived variable or constant stack-resident and storage-pool-resident objects.

2.2 Storage Pool Reclamation

In Ada, storage pools can be automatically reclaimed when the scope of the associated access type is exited. This will happen naturally if the storage pool is represented in the PSVM by a vector allocated at the point of the declaration of the access type. However, this poses a challenge for identifying storage pools when using a general access type, because any sort of global table of storage pool vectors will have elements that go out of scope. Because of Ada's Unchecked_Access capability, it is possible that a general access value might designate an object in a storage pool that no longer exists.

The PSVM has an existing mechanism for handling the reclamation of storage. The PSVM allocates objects within storage regions, which play a role similar to Ada's storage pools, but are heterogeneous in that they contain all types of objects allocated within a given scope. When the scope exits, the storage region is reclaimed, making the memory available for reuse by other storage regions. Storage regions are assigned unique integer identifiers, and these identifiers are available for reuse upon exit from the region's scope.

Because of the ownership-based storage model used by the PSVM, there is no danger that a reference to a storage region will outlive the storage region, but we need to allow for this possibility when mapping Ada to the PSVM. One well-known way to handle this is to associate a *generation number* with each reference, and whenever a storage region is reclaimed, the generation number associated with its integer identifier is incremented. Any use of a storage region identifier must check the generation number as well, if there is some possibility that the region has been reclaimed, as would be the case for an Ada general access type. Therefore, our mapping for an Ada general access type will not only need to identify an individual object within the storage region where its storage pool resides, but will also need to include the particular generation number for the storage region, in case the access value is a *dangling* reference that might have been created using Unchecked_Access.

2.3 Individual Object Reclamation

Both in Ada and in the PSVM, individual objects may have their storage reclaimed, even while their associated storage pool/region remains in scope. In Ada, this is accomplished with a call on an instance of Unchecked_Deallocation. This sets the given pointer to null after freeing storage for the designated object, but other references might remain, which thereby become dangling references. In the PSVM, reclamation happens automatically when an object is set to the null value for its type, but because of ownership-based semantics, there will not be any other references, and so there are no dangling references to worry about.

To provide a level of safety when mapping Ada to the PSVM, we need to introduce generation numbers for individual object identifiers as well. That is, even for a pool-specific access value, the index in the storage pool vector must have an associated generation number, which is incremented when Unchecked_Deallocation is applied. Any

use of a storage pool index must check this generation number, before indexing into the associated vector.

The net effect of having these two distinct generation numbers is that a general access value ends up with four components: a storage-pool (or storage-region) identifier, a pool/region generation number, an index (*slot number)* into the storage vector, and an individual element generation number.

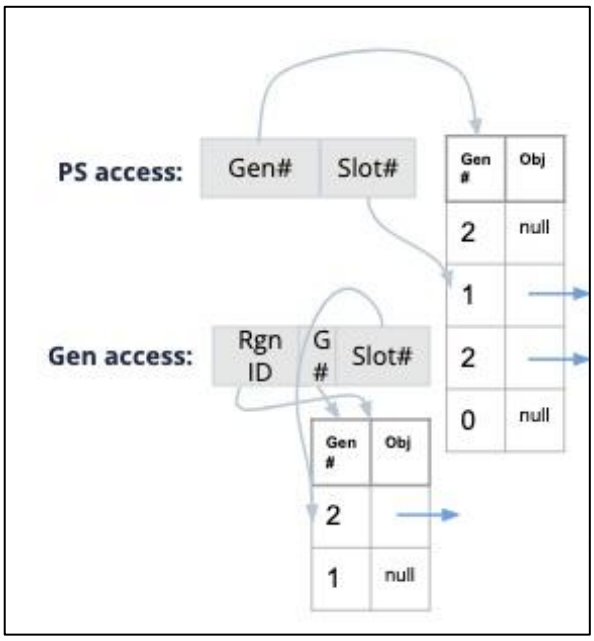


Figure 2 Representation for Pool-Specific and General Access Values

In Figure 2, we illustrate our initial suggested representation for Ada access values in the PSVM. The pool-specific access value has two components, a generation number and a slot number. The access value is considered a dangling reference if the generation number in the access value does not match the generation number stored at the given slot. If the generation numbers match, then the object at the given slot is the designated object for the access value.

For a general access value, a pool/region identifier (Rgn ID in the figure) is used to find the table of objects, and then the generation number and slot number are used as for the pool-specific access value. As indicated above, the pool/region identifier also has a generation number, which if it doesn't match, the whole pool/region has been reclaimed, and again, the access value is considered a dangling reference.

2.4 Unified Access Value Representation

For uniformity, we ultimately adopted the representation used for general access values, for pool-specific access values as well. However, for pool-specific values, there is no need to look at the pool/region information, since the particular pool/region is known at compile time. Using the same representation also simplifies the process of converting from a pool-specific access value to a general access value, because the access value already has all four components filled in.

In Figure 3, we illustrate the unified representation for access values on the PSVM. We also illustrate the choice of combining all the storage pool vectors into a single vector of aliased objects for a given PSVM storage region. This allows us to reuse the PSVM region index to identify the region associated with the object designated by a general access value, and to use a per-storage-region slot number to identify any aliased object allocated from a given PSVM region.

To ease checking generation numbers, we now give every slot in a storage-region vector a *self address*. Before using an access value, we compare it against its self address, and if they do not match, the access value must be a dangling reference. This effectively combines the checks for region and slot generation numbers, and avoids the need to isolate the generation numbers before performing the check.

Combining all the storage pool vectors into one vector per PSVM storage region also answers the question of how we should handle stack-resident aliased objects. We simply assign them a slot within the storage region with which they are already associated, and insert the implicitly-dereferenced access value in their place. As an optimization, we could postpone assigning a slot to a stack-resident aliased object until an access value is created that designates it, though that implies more complexity and synchronization when evaluating an Access or Unchecked_Access attribute.

2.5 Performance and Safety Issues

One inevitable effect of this index-based representation for access values is the increase in overhead of dereferencing an access value. As indicated above, pool-specific values are somewhat simpler, since the associated storage region is

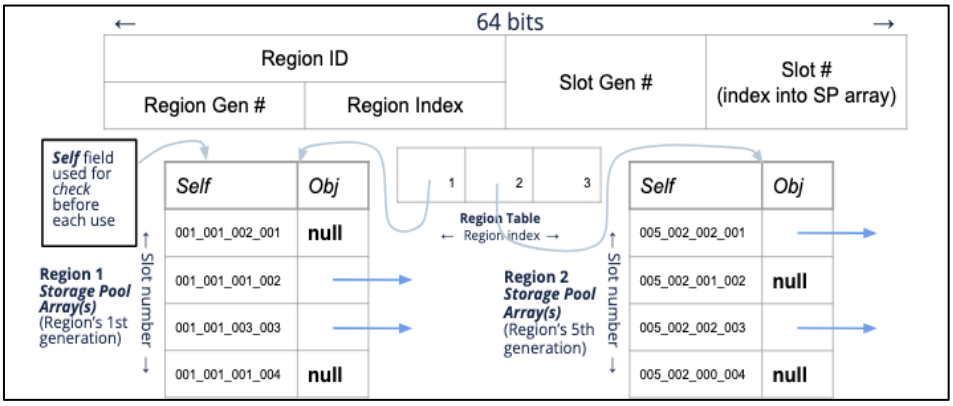


Figure 3 Unified Representation of Access Values on the PSVM

known at compile time, and so a dereference requires only using the slot number to index into the storage region's vector of aliased objects, and then checking the self address, before using the object at the given slot. For a general access value, we must first use the region index to identify the storage region, and then index into its vector of aliased objects using the slot number, and then, again, check the self address.

Other operations with access values such as conversion and deallocation do not incur significant extra overhead. As mentioned above, using the Access or Unchecked_Access attribute to create an access value designating a stack-resident object might incur overhead if the slot number has not been allocated when the object was created. If it was assigned then, the Access or Unchecked_Access involves merely fetching the implicitly-dereferenced access value that is already in place of the original aliased object.

The additional overhead on dereferencing does provide additional safety, in that Ada normally does not detect uses of dangling references. With this mapping to the PSVM, it is straightforward to detect any use of a dangling reference, because of the self address checks performed.

The ParaSail language also takes advantage of the ownership-based storage model to detect data races, preventing multiple threads from potentially making conflicting references to a single object. This additional safety property turns out to depend on compile-time checks that detect inappropriate attempts to create multiple references to the same object. Because this is work done by the compiler rather than the underlying PSVM, this additional race detection does not come over to Ada automatically by using the PSVM. However, Ada 2022 includes its own mechanisms for detecting conflicting references to shared data, with one checking mode that is quite similar to that used in ParaSail (All_Parallel_Conflict_Checks, see [1] §9.10.1), so this safety property can be preserved for Ada.

3 Supporting exceptions with parallel threads

The other Ada features which present a challenge when implementing on top of the ParaSail virtual machine are exceptions and exception handling. The ParaSail virtual machine has no direct equivalent to exception handling. However, as mentioned it directly supports lightweight parallelism. In this section we will examine the various options for supporting exception handling on top of such a virtual machine, and what are the implications for safety and performance.

Ada exceptions provide a basic capability of raising an exception and having it handled by the nearest dynamically enclosing exception handler. In addition, some amount of information can be communicated along with the exception occurrence, which can include a user-supplied string, presumably indicating something about the cause of the exception.

If an Ada task raises an exception but there is no handler within the task, it terminates the task, optionally invoking a termination handler first. On the other hand, for the lightweight logical threads of control introduced by Ada 2022, an exception raised in such a thread, if not handled directly within the thread, will attempt to propagate the exception to the invoking thread, while attempting to terminate all other sibling threads. If multiple subthreads concurrently attempt to propagate an exception, one of them is chosen arbitrarily, and the exception propagated by that subthread is propagated to the invoking thread.

The PSVM has no direct support for exceptions, but as illustrated in Figure 4, it does have a number of threading operations, based on the notion of *masters* each with a set of *subthreads*. Start_Parallel spawns a subthread and an associated master, Wait_For_Parallel waits for all the subthreads of a given master, and Add_Parallel_Call adds another subthread to a given master as part of performing a function call. In addition, the PSVM has a mechanism for a subthread to *exit* in a way that indicates that it has completed the job that was being performed by that subthread and its sibling subthreads, thereby terminating all of the subthreads of the given master, and providing a result value back to the invoking thread. Similarly to the Ada 2022 exception semantics, if there are multiple subthreads that attempt such a preemptive exit concurrently, one is chosen arbitrarily and the other subthreads attempting to exit are terminated.

Mapping the Ada exception handling mechanism to these PSVM threading operations turns out to be relatively straightforward. All we need to do is to create a new master when we enter a handled sequence of statements, with a flag indicating that this master is an exception handler. When an Ada exception is raised by a given thread, we simply look back up the tree of masters until we find one that has the exception-handler flag set, and perform the equivalent of the "exit" operation to that master, providing the identity of the exception and optionally a message as the "result" provided by the subthread. When the master is notified of such an *exceptional* exit, it invokes the statements of the exception handler. A normal completion of the subthread(s) of the

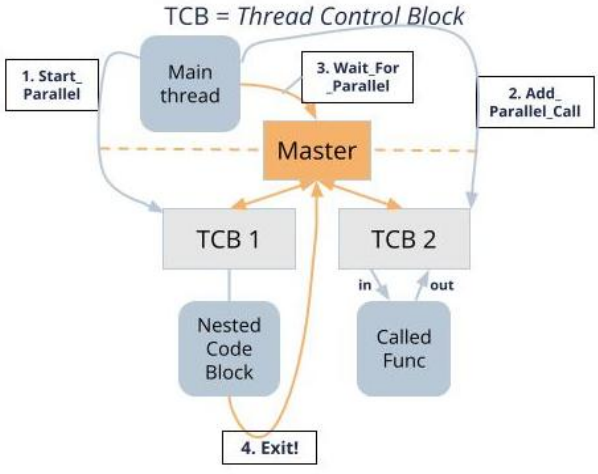


Figure 4 PSVM Threading Operations

master will result in continuing the normal control flow. The existing PSVM mechanism for resolving race conditions between concurrent threads invoking an "exit" operation, provides the corresponding resolution of multiple Ada threads concurrently propagating exceptions.

4 Conclusions

A safe virtual machine that supports ownership-base storage and lightweight parallelism, such as the ParaSail Virtual Machine, can provide a robust basis for an Ada 2022 implementation, while providing some additional safety guarantees. Safety that is dependent on compiler checks will still need corresponding checks in the Ada compiler, but certain checks such as for dangling references can be provided directly as part of the mapping to the ownership-based storage model of the virtual machine.

One somewhat surprising conclusion of this effort was that, given an appropriate set of lightweight threading primitives, exception handling can be handled by a virtual machine that does not have direct support for exceptions. This initial effort of mapping Ada 2022 features indicates that the ParaSail virtual machine could be a good platform for evaluating future proposals for extensions to the Ada language.

References

[1] R. Duff, S. T. Taft, et al, Ada Reference Manual, *http://www.ada-auth.org/standards/22rm/html/RM-TOC.html*, 2002.

[2] S. T. Taft, "ParaSail: A Pointer-Free Pervasively-Parallel Language for Irregular Computations", *The Art, Science, and Engineering of Programming*, Vol. 3, Issue 3, Article 7, https://arxiv.org/abs/1902.0052, 2019.

Software Verification and Generative AI – Some Practical Examples and Considerations

M. Martignano
Spazio IT – Soluzioni Informatiche s.a.s – https://spazioit.com.

A. Damiani, L. Nucciarelli
GemelliGenerator – Fondazione Policlinico Universitario Agostino Gemelli – https://gemelligenerator.it

D. Gui, S. Magalini
Università Cattolica del Sacro Cuore – Campus Roma – https://roma.unicatt.it/facolta/medicina-e-chirurgia

Abstract

Software verification, that is requirements baseline analysis, technical specification analysis, design analysis and code and testing analysis [1], is a crucial aspect of software development, ensuring that the products of each development phase satisfy the conditions imposed at the start of that phase [2]. Traditional software verification techniques often rely on manual effort, which can be time-consuming and error prone. However, with recent advancements in Generative Artificial Intelligence (AI) and Large Language Models (LLMs), there is a growing opportunity to automate and improve software verification activities. This paper describes how Generative AI, particularly LLMs, can facilitate software verification activities, including understanding of documentation, code analysis, bug detection and testing. Benefits are presented together with the associated challenges and limitations, especially the potential risk of exposing sensitive and proprietary information.

Keywords: Generative AI, LLM, SLM, Software, Verification.

1 Introduction

Software verification, according to the ESA ISVV Handbook [1], is divided in four macro areas: requirements baseline analysis, technical specification analysis, design analysis and code and testing analysis. Many of the activities belonging to these areas require verifying the correctness and completeness as well as the internal and external consistency of the item under analysis, which is either a piece of documentation or a piece of source code. Therefore, the capability of understanding documentation, i.e. written text, and source code is vital for the proper execution of software verification.

In recent years, Generative Artificial Intelligence (AI) and Large Language Models (LLMs) have emerged as powerful tools for automating various tasks in software development. LLMs, such as OpenAI's GPT (Generative Pre-trained Transformer) models, have demonstrated remarkable capabilities in terms of analysing and understanding natural language and source code. Leveraging the capabilities of Generative AI for software verification activities holds the potential to improve efficiency, accuracy, and scalability.

When analysing documentation, AI systems could check documents like requirements, technical specifications, user manuals, API documentation to identify inconsistencies, errors or ambiguities and assuring their completeness and accuracy. Likewise, when analysing source code, AI systems could check the formatting and style, identify anomalies like potential bugs, security vulnerabilities, performance issues and suggest fixes. On top of that, AI systems could also check the consistency at near-semantic level between a piece of document and a piece of source code, as shown in Section 4.

Despite the potential benefits of leveraging Generative AI for software verification, several challenges and limitations must be addressed, e.g.: the produced results could be incorrect (artificial hallucination), AI models may inadvertently propagate biases present in training data or generate unethical code; understanding and interpreting the decisions made by Generative AI models can be challenging, particularly in complex software verification tasks; Generative AI models often require significant computational resources and time for training and inference; effective integration of Generative AI into software verification workflows requires seamless collaboration between human experts and AI systems; handling sensitive codebases and proprietary software information raises concerns about data privacy and security.

This paper is organized in seven sections. The first one is this introduction. The second one briefly describes Generative AI, LLMs. The third one shows some practical examples of applying Generative AI to document analysis. The fourth one shows some practical examples of applying Generative AI to source code analysis. The fifth one presents the challenges and limitations associated with using Generative AI for software verification. The sixth one concentrates only on the problem of handling sensitive codebases and proprietary software. The seventh one will try to imagine the future of software verification and show how it will be shaped by advancements in generative AI models,

integration with existing verification tools, and community-driven initiatives for responsible AI development.

2 Generative AI and Large Language Models

Generative AI and Large Language Models (LLMs) are related but distinct concepts in the field of artificial intelligence. The term Generative AI refers to any AI system whose primary function is to generate content. This could include a variety of AI models that generate different types of content, such as images, text, code, audio and video. Generative AI emphasizes the content-creating function of these systems. On the other hand, Large Language Models (LLMs) are a specific type of AI system that works with language. They are designed to understand and produce text. LLMs are a form of generative AI, but they specifically deal with text-based content. Some notable LLMs are OpenAI's family of GPT [3] (Generative Pre-trained Transformer) models – both OpenAI ChatGPT and Microsoft Copilot are based on GPT; Google's PaLM [4] (Pathways Language Model) and Gemini [5]; Meta's LLaMA [6] (Large Language Model Meta AI) open-source models.

The landscape of open-source Language Models is diverse [7], ranging from large to small in terms of parameter count, and from generic to domain-specific in their applications. Indeed, Small Language Models (SLMs) refer to versions of language models that have fewer parameters compared to larger models. These models are designed to be computationally lighter and more resource-efficient, making them suitable for deployment in constrained environments such as mobile devices, IoT devices, or edge computing systems where computational resources are limited. The selection of an open-source model provides developers with the flexibility to deploy it on a computing platform of their choice, be it cloud-based or on-premises. Further details on this topic can be found in Section 6.

3 Examples of Documents Analysis

In these experiments ChatGPT has been compared with two open-source models, i.e. Salesforce's XGen-7B-8K-Inst [8] and Mistral AI's Mistral AI's Mixtral-8x7b [12]. It is worth noting that all open-source models are characterized by two figures: the number of parameters (in this case 7B) and the length of the input sequence (in this case 4K). While ChatGPT (GPT-3.5 the free version) was obviously executed on the cloud, XGen was executed on a local computing platform – a gaming laptop with an Intel i7-10875H CPU @ 2.30GHz CPU, 32GB of RAM and an NVIDIA GPU RTX 280 GPU. For each experiment the same prompt was submitted to the three models. The full versions of both the prompt and the responses from the models can be found on Spazio IT website [9]; only the commentary is provided here.

3.1 Kafka's Before the Law (ex1)

In this experiment the three models have been asked to summarize Kafka's short story "Before the Law" [10]. All models answered with a proper summary; ChatGPT answer was a bit more concise and more fluent. The point is that also the open-source models, executed on a local computing platform, were able to produce a good result; actually, Mixtral produced the best summary, also containing some possible interpretations of Kafka's story.

3.2 Poorly written requirements (ex2)

In this experiment, a text containing four inadequately formulated requirements was presented to the models. The first and third requirements were excessively ambiguous, while the second wasn't a requirement in the slightest. The final requirement pertained to multiple systems. ChatGPT only flagged the first and third requirements, which were overly vague. XGen only pinpointed the second requirement, which wasn't a requirement at all. Also in this case, Mixtral produced the best result, showing a correct understanding of the submitted text.

4 Examples of Code Analysis

In these experiments ChatGPT has been compared with open-source Meta's CodeLlama-70b-Instruct-hf [11] and Mistral AI's Mixtral-8x7b-instruct [12]. The open-source models were executed via the Perplexity AI GUI [13] presumably on AWS P4d instances, which are powered by NVIDIA A100 Tensor Core GPUs. For each experiment the same prompt was submitted to the three models. Also in this case, the full versions of both the prompt and the responses from the models can be found on Spazio IT website [9]; only the commentary is provided here.

4.1 Conversion Error in C (ex3)

In this experiment, a tiny snippet of code was tested on the three different models. The code snippet involved assigning a negative value (-1) to an unsigned integer variable. All three models successfully identified the issue. CodeLlama offered the most concise explanation, while ChatGPT provided a more detailed explanation focusing on the specifics of the C language. Mixtral's explanation delved into both the C language and the data representations of integers.

4.2 Homomorphic Encryption Example (ex4, C++17)

In this experiment, a compact C++17 application was presented to the models. This application, which performs a tabular search on homomorphically encrypted data, is built on the IBM HElib library [14]. Both ChatGPT and Mixtral demonstrated comprehension of the application and provided positive feedback. Furthermore, Mixtral offered some recommendations for further enhancements. In contrast, CodeLlama appeared to misunderstand the application, providing a negative evaluation and some generic, unrelated suggestions such as avoiding SQL injection.

4.3 Dining Philosophers (ex5, C++20)

In this experiment, the models were tasked with evaluating whether the classic "Dining Philosophers" concurrency problem was addressed by the C++20 Dijkstra implementation featured on Wikipedia [15]. All three models demonstrated an understanding of both the problem statement and the C++ implementation. CodeLlama's

assessment was entirely positive. ChatGPT also gave a positive review but emphasized that the implementation requires thorough testing and validation. Mixtral identified potential for enhancing the fairness of the synchronization mechanism and offered some recommendations. This experiment serves as an illustration of the near-semantic relationship (traceability) between textual content and source code.

4.4 Dining Philosophers (ex6, Ada95)

In this study, the models were assigned to assess if the renowned "Dining Philosophers" concurrency problem was tackled by the Ada95 code authored by Michael B. Feldman [16]. Only ChatGPT and Mixtral showed comprehension of both the problem description and the Ada95 implementation, and they positively evaluated the structure of the code. Additionally, Mixtral pinpointed a potential issue in the synchronization mechanism that was implemented. Also this experiment serves as an illustration of the near-semantic relationship (traceability) between textual content and source code.

5 Challenges and Limitations

This Section attempts to address the challenges and limitations mentioned in the introduction.

5.1 Artificial hallucinations (ex7)

In AI the term “artificial hallucinations” refers to inaccurate or misleading output generated by AI models, especially LLMs. These hallucinations can manifest in different ways, as false or invented information; misinterpretations and biases; identifying patterns where there are none. Several factors contribute to these hallucinations, including insufficient training data, incorrect assumptions, and adversarial attacks. The implications and applications of artificial hallucinations include risks in tasks requiring accuracy, such as medical diagnosis [17], and the potential for spreading misinformation. However, controlled hallucinations can have creative potential in art and can help explore future scenarios or uncover hidden connections in complex data. Hallucinations may occur also when analysing code: errors in the model reasoning could bring to the identification of false positives. Some few examples of these “hallucinations” can be found on Spazio IT website [9], ex7.

5.2 Bias propagation

The term “bias propagation” in AI describes the process where biases, inherent in the data used for training machine learning models, are carried forward or even intensified in the model’s results. This happens when the data used for training is biased, for instance, societal biases or imbalanced representation of certain groups, which the model then assimilates and mirrors in its forecasts or choices. Various mechanisms can cause bias propagation, such as: data bias, which is the bias already existing in the training data; algorithmic bias, where certain algorithms might inherently favour or strengthen certain biases; feedback loop, where AI-based predictions or decisions can influence future data gathering or human behaviour, thus creating a feedback loop; contextual bias, where the AI might overlook the wider context or historical biases when making decisions, leading to biased results. Bias propagation in AI can have considerable ethical and societal consequences, as it can lead to unjust treatment or discrimination against certain groups. Bias may occur also when analysing code: a model trained using commercial / administrative software may be not suitable for analysing avionics software.

5.3 Unethical text/code generation

Unethical text or code generation in AI refers to the creation of content or software by artificial intelligence systems that can cause harm, violate ethical principles, or infringe upon societal norms. This unethical behaviour can manifest in different ways: malicious content generation: AI systems can be trained to generate malicious or harmful content; infringement of privacy: AI-generated content may infringe upon individuals' privacy rights by generating personal information or sensitive data without consent; promotion of hate speech or discrimination: AI-generated text or code may propagate hate speech, discriminatory language, or biased algorithms; creation of dangerous software: AI-generated code may result in the creation of software or algorithms that pose risks to cybersecurity, public safety, or critical infrastructure; unfair bias and discrimination: AI-generated content or code may reflect and perpetuate biases present in the training data, leading to unfair treatment or discrimination based on characteristics such as race, gender, or socioeconomic status. Tackling the issue of unethical text or code generation in AI necessitates a thoughtful examination of ethical norms, regulatory structures, and conscientious AI practices. This involves the establishment of protective measures to avert the creation of damaging content, advocating for transparency and responsibility in AI creation and implementation, and encouraging cooperation among stakeholders to lessen risks and advance ethical AI innovation.

5.4 Explicability

Explicability in AI, also known as explainability or interpretability, refers to the ability to understand and interpret how an AI system arrives at its decisions or outputs. It involves making AI systems transparent and understandable to humans, particularly to stakeholders such as end-users, developers, regulators, and policymakers. Explicability is essential for several reasons: trust, users need to trust AI systems and understanding how they work fosters trust; accountability, when AI systems make decisions that impact individuals or society, it's crucial to be able to trace and understand those decisions for accountability purposes; compliance, some regulations require explanations for automated decisions, such as the GDPR's "right to explanation"; debugging and improvement, interpretability facilitates debugging and improving AI systems by identifying errors or areas for enhancement. In the Dining Philosophers experiments, ex5 and ex6, the Mixtral-8x7b-instruct model was tasked with providing explanations for its evaluations.

5.5 Computational resource and Quantization

Large language models demand huge computational resource. To be able to run these models on local and affordable computing platforms a technique, called "quantization" is used. Quantization refers to the process of reducing the precision of model parameters, typically by converting floating-point numbers (which have decimal precision) into integers with a smaller bit-width representation. This process helps in compressing the model size and reducing computational complexity, making it more efficient for deployment on resource-constrained devices or for faster inference. Quantization is a crucial technique for deploying large neural network models like LLMs in real-world applications where computational resources are limited, such as mobile devices or edge devices. It enables efficient inference while minimizing the impact on model accuracy. However, quantization may also introduce some loss of accuracy, so careful tuning and optimization are necessary to balance model size, computational efficiency, and performance.

In addition to streamlining or "downsizing" the model, another method of conserving resources is to utilize efficient software platforms or frameworks, such as "ggml" [18]. It's widely recognized that some programming languages outperform others in terms of speed, efficiency, resource usage, and carbon footprint [19].

5.6 Integration of AI in the ISVV workflows

All activities in ISVV involve the examination of extensive document sets, which often comprise lengthy documents, and large codebases made up of numerous files. All Large Language Models have a limit to the size of prompts (the input sequence) they can accept, which is expressed in terms of the number of tokens. Consequently, the text and code submitted to a model cannot exceed its input size. However, this is not a significant issue in ISVV as all technical documentation is typically organized into easily identifiable sections. Similarly, a large codebase can be broken down into smaller code segments using some form of static analyser/compiler.

Generative AI instruments can assist human evaluators in comprehending and scrutinizing both documentation and code at a near-semantic level. This introduces fascinating opportunities such as validating the traceability among documents and/or between documentation and source code at a near-semantic level, surpassing mere lexical analysis.

6 Handling sensitive codebases and proprietary software

When users utilize generative AI systems like OpenAI's ChatGPT, Google's Gemini, or others provided by major tech companies for free, there's no assurance that the data they input, i.e. the Content, specifically the information within the prompts, remains confidential. In fact, as an example in ChatGPT "Terms of Use" it is stated: "Our Use of Content. We may use Content to provide, maintain, develop, and improve our Services, comply with applicable law, enforce our terms and policies, and keep our Services safe."

Certainly, it is always feasible to set up commercial registrations or contractual agreements that prohibit the service provider from retaining and exploiting user content, as mentioned in the below statement, always in ChatGPT "Terms of Use".

"Opt Out. If you do not want us to use your Content to train our models, you can opt out by following the instructions in this Help Center article. Please note that in some cases this may limit the ability of our Services to better address your specific use case."

Unfortunately, these are merely business assurances. The sole method for users to ensure their content isn't misused is by exclusively utilizing open-source models (with a thorough understanding of their functions) and operating these models on a private computing platform.

7 Software Verification and Generative AI – Future directions

This paper has provided an overview of various perspectives and considerations, all converging on a shared foundation that delineates what merits attention in the imminent integration of Software Verification with Generative AI. This common foundation is encapsulated by the following assertions:

1. Generative AI stands poised to enhance Software Verification by facilitating analysis and comprehension at a near-semantic level for both documents and source code.
2. Open-source models are deemed preferable, if not essential, to address privacy concerns effectively.
3. Similarly, in addressing privacy issues, the adoption of privately owned computing platforms is advocated, if not mandated.
4. To mitigate resource consumption, it is advisable to employ quantized models operating on efficient software platforms.

Acknowledgements

This research initiative is supported by the European Union's Horizon Europe Framework Programme for Research and Innovation, under the TRUSTEE project (Grant Agreement No. 101070214). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] European Space Agency – ESTEC, "Critical Software S.A. and Rovsing A/S", *Independent Software Verification and Validation Handbook,* ref. CSW-ESAISVV-2022-GBK-02897, 2022.

[2] IEEE, "IEEE Standard for System, Software, and Hardware Verification and Validation", in *IEEE Std 1012-2016* (Revision of IEEE Std 1012-2012/

Incorporates IEEE Std 1012-2016/Cor1-2017), pp.1-260, 2017.

[3] OpenAI, *Introducing GPTs*, https://openai.com/blog/introducing-gpts.

[4] Google, *Google's Palm*, https://ai.google/discover/palm2.

[5] Google, *Google's Gemini*, https://deepmind.google/technologies/gemini/.

[6] Meta, *Meta's LLaMa,* https://llama.meta.com/.

[7] HuggingFace, *Some Open-Source Models* https://huggingface.co/models.

[8] Salesforce, *Salesforce 's XGen-7B-8K-Inst* https://huggingface.co/Salesforce/xgen-7b-8k-inst.

[9] *Spazio IT Experiments with Language Models* https://spazioit.com/pages_en/sol_inf_en/experiments-with-language-models/.

[10] *Kafka's Before the Law* https://www.kafka-online.info/before-the-law.html.

[11] HuggingFace, *Meta's CodeLlama-70b-Instruct-hf* https://huggingface.co/codellama/CodeLlama-70b-Instruct-hf.

[12] HuggingFace, *Mistral AI' Mixtral-8x7B-Instruct* https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1.

[13] Perplexity, *Perplexity AI GUI*, https://labs.perplexity.ai/.

[14] IBM, *IBM Helib*, https://github.com/homenc/HElib.

[15] *Dining Philosophers in C++20*, https://en.wikipedia.org/wiki/Dining_philosophers_problem.

[16] *Dining Philosophers in Ada95*, https://adalang.io/docs/style-guide/Portable_Dining_Philosophers_Example/.

[17] *H.* Alkaissi, S. I. McFarlane, "Artificial Hallucinations in ChatGPT: Implications in Scientific Writing", *Cureus* 15(2): e35179, 2023.

[18] ggml.ai, *ggml*, https://ggml.ai/.

[19] *R.* Pereira, M. Cuoto, F. R. Rui Ra, J. Cunha, J. P. Fernandes, J. Saraiva, "Energy Efficiency across Programming Languages. How Do Energy, Time and Memory Relate?", 1*0th ACM SIGPLAN International Conference*, 2021.

AI Augmented Requirements Engineering in the AIDOaRT Project: NLP Techniques and Language Models to Encode Requirements Text Semantically for the Railway Industry

Alessandra Bagnato

Softeam, Softeam Software, Docaposte Groupe, 3 avenue du Centre 78280 Guyancourt.; Tel : +33 1 30 12 18 58; e-mail: Alessandra.bagnato@docaposte.fr

Bilal Said

Softeam, Softeam Software, Docaposte Groupe, Nantes, France; e-mail : bilal.said@docaposte.fr

Abstract

Requirements engineering (RE) remains a complex and crucial preliminary phase in the life cycle of a Cyber-Physical System (CPS) design and development process. In CPS, such as railway and automotive systems, the requirements of a single system, often expressed and handled in unstructured text format, may exceed several thousands of clauses, with technical jargon, multitude of references to external safety and functional standards, extensive cross-references and dependencies. This makes RE activities, such as requirements analysis, validation and allocation, time consuming and error prone for system engineers. This paper contributes with Natural Language Processing (NLP) Techniques and Language Models (LM) to encode requirements text semantically. It describes NLP4RE, a tool by Softeam for requirements semantic similarity search. It has been applied for a use case from the railway industry within the AIDOaRT project. In this use case, system engineers get valuable insights on new bid documents by retrieving semantically similar clauses from previous projects.

Keywords: Requirements Engineering, AI, ML, Natural Language Processing, Language Models, Model-Driven Engineering.

1 Introduction

Requirements are often elicited in natural language and express customers' input mainly with text, images, tables, and external references. Thus, requirements analysis is a time-consuming, error-prone, and human labour-intensive task. In the complex system industry in particular, each customer project involves thousands of hardware and software requirements, frequently using very detailed technical and legal terms.

Recent advances in AI/ML and Natural Language Processing (NLP) techniques may provide valuable insights to requirements engineers. They may help to increase accuracy and reduce the time needed to process new project requirements based on prior knowledge mined from previous projects. However, most techniques are developed and validated on relatively small datasets.

This paper aims to present the NLP4RE Prototype, developed by Softeam R&D team, and capable of investigating and finding semantic similarities between new requirements and requirements from previous projects to then help to create an automated solution to enhance the engineers' analysis.

The NLP4RE Prototype is designed for use in requirements engineering case studies where businesses and industries are studying recurrent clients' bids or tenders with relatively similar or standard requirements specification documents and who are interested in providing their RE teams with insights on new projects based on responses to previous ones. The prototype can also provide help applying the NLP4RE semantic search in large software and system models (e.g., UML, SysML…), as well as wide organisation and enterprise architectures (e.g., Archimate, TOGAF), to easily find similar model elements, for instance requirements, given their description. The prototype uses Natural Language Processing techniques and language models to encode requirements text semantically and helps requirements engineering teams while studying new bid and tender documents by identifying similar requirements from previous projects, and thus to tackle and answer the new requirements based on answers to the previous similar ones (e.g., allocate similar requirements to similar teams, respond to requirements compliance similarly).

The AI Augmented Requirement Engineering achieved will help to minimise the time spent analysing the requirements and maximise the number of correctly evaluated and answered requirements.

The work described is done by the Modelio team in the context of the AIDOaRT project [1] which focuses on AI-augmented automation supporting modeling, coding, testing, monitoring, and continuous development in Cyber-Physical Systems (CPS). The project proposes to combine Model

Driven Engineering principles and techniques with AI-enhanced methods and tools for engineering more trustable CPSs.

Within the AIDOaRt project, a use case proposed by the railway industrial partner Alstom [8] aims, among other objectives, to automate and improve their requirements engineering process using solutions that would analyse requirements leveraging the advantages of Artificial Intelligence techniques. In AIDOaRt, Alstom has collaborated with other research partners such as Mälardalen University (MDU), Research Institutes of Sweden (RISE) and Softeam [2][3] to include AI Augmented Requirement Engineering in their current tasks [4]. While MDU proposed methods for requirements ambiguity check, and RISE tackled the problem of requirements allocation, we focused on requirements semantic similarity check.

This paper presents Softeam's NLP4RE prototype and contributes towards improving the state of the art in the analysis of critical requirements, customer specification and use the AI to get recommendations on requirements similarities for suitable actions with first tests done within the Alstom context [8].

2 The AIDOaRT Project

Funded by the European Union AIDOaRt is a 42 months long H2020-ECSEL European project involving 31 organisations belonging to both the academic and the industrial world, grouped in clusters from 7 different countries (Sweden, Finland, Italy , France, Czech Republic, Austria, Spain) focusing on AI-augmented automation supporting modeling, coding, testing, monitoring, and continuous development in Cyber-Physical Systems (CPS). AIDOaRT began on 1st April 2021 for a period of 42 months.

The AIDOaRT aims to create a framework incorporating methods and tools for continuous software, system engineering and validation, leveraging the advantages of AI techniques (notably Machine Learning) to provide benefits in significantly improved productivity, quality, and predictability of CPSs, CPSoSs, and, more generally, large and complex industrial systems. The AIDOaRt project approach includes AI-augmented automation supporting modeling, coding, testing, monitoring, and continuous development in Cyber-Physical Systems (CPS).

On the academic side, it includes Åbo Akademi University - Finland (ABO), Mälardalen University - Sweden (MDU), RISE Research Institutes of Sweden - Sweden (RISE), AIT Austrian Institute of Technology GMBH - Austria (AIT), Graz University of Technology - Austria (TUG), Johannes Kepler University Linz - Austria (JKU), Brno University of Technology - Czech Republic (BUT), Institut Mines-Telecom - France (IMTA), University of L'Aquila - Italy (UNIVAQ), University of Sassari - Italy (UNISS), Universitat Oberta de Catalunya -Spain (UOC), University of Cantabria -Spain (UCAN).

From the industrial use case side, it includes: Abinsula SRL - Italy (ABI), AVL List GMBH - Austria (AVL), Alstom AB - Sweden (BT), CAMEA SPOL SRO - Czech Republic (CAMEA), Clearsy SAS - France (CSY), HI Iberia Ingeniería Y Proyectos SL - Spain (HIB), Prodevelop - Spain (PRO), TEKNE SRL - Italy (TEK), Volvo Construction Equipment AB - Sweden (VCE), Westermo Network Technologie - Sweden (WESTMO).

From technology providers side it includes Softeam - France (SOFT), Automated Software Testing GmbH - Austria (AST), Dynatrace Austria GmbH - Austria (DT), Anders Innovations - Finland (AND) , Qentinel Oy - Finland (QEN), Intecs Solutions SPA - Italy (INT), Ro Technology SRL - Italy (ROTECH) , ACORDE Technologies SA - Spain (ACO), ITI Instituto Tecnologico de Informatica - Spain (ITI).

The industrial use case partners in AIDOART provide industrial case studies for the project. Each case study comes with a set of requirements and a set of existing technologies and tools which are offered to the other two types of partners as a set of baseline technologies and potential exploitation assets to experiment with and to develop on

Both, the technology providers and the research partners develop new technologies and tools or improve the current technologies to satisfy the requirements specified by the industrial use case partners in close collaboration with them.

3 Typical CPS Requirements Engineering (RE) Process and AI-Augmented Methods

CPS design and development industries, such as railway and automotive systems constructors, receive thousands of customer text requirements in each new project. Engineers must accurately evaluate and answer these requirements one by one [4]. The engineers must allocate them to different teams and analyse whether the CPS manufacturer can comply with the requirements. The analysis of the requirements is thus extensive and time-consuming [5]. A large panel of solutions uses NLP techniques to automate and assist engineers in these requirements engineering tasks [10, 11].

For the AIDOaRt tackled use case, MDU has proposed an NLP requirements ambiguity checker [9], and RISE checked for software and requirements similarity correlation [12]. We, at Softeam, focused on requirements semantic similarity search [6, 13]. The dataset proposed by our industrial partner is composed of 5 different projects, spanning over several years, and with thousands of requirements each. This allowed us to evaluate the relevance of the search for recent project requirements in previous ones. Syntactic similarity would have been unsuccessful in detecting complex semantic relationships between textual requirements. In fact, it is impossible for a keyword or term based search to identify that a requirement stating that "Under extreme temperatures, the unit must [...]" is similar to another specifying that "Below -18° C and above 35° C, the unit must [...]". Accordingly, more elaborate semantic-based similarity search methods [14] must be developed and used in order to provide useful insights to requirements engineers.

4 Softeam's Proposed Requirements Semantic Search

Within the AIDOaRt context, the Modelio R&D team of Softeam [2, 3] tackled these RE challenges using various NLP and AI/ML techniques and developed the NLP4RE Prototype to allow automatic requirements identification, extraction, classification and similarity search in textual documents [6, 13]. In this paper, we present the "Requirements Semantic Search" capability (cf. Figure 1).

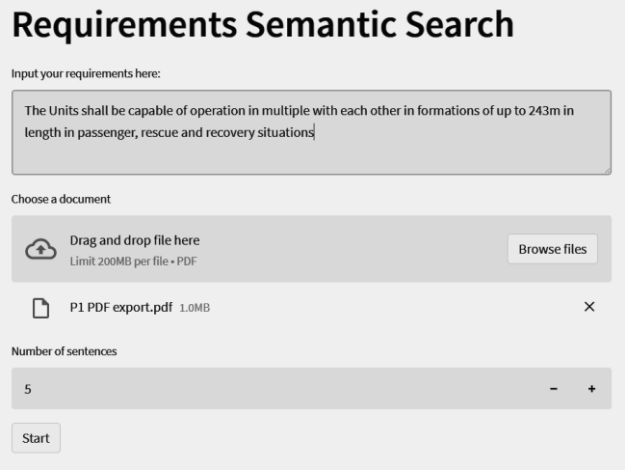


Figure 1. The user interface of the "Requirement Semantic Search" capability in the NLP4RE Prototype

Given a requirement from a new project, we display in our proposed prototype the top *k* (e.g., 5) most similar requirements from old projects (cf. Figure 3). Furthermore, we augment the displayed results with highlights of the common lexical similarities to provide a potential explainability (cf. underlined words in Figure 2). This interaction between the tool and the user is meant to ensure the implementation of a responsibly designed tool that explains and justifies its decisions in an easy-to-understand manner.

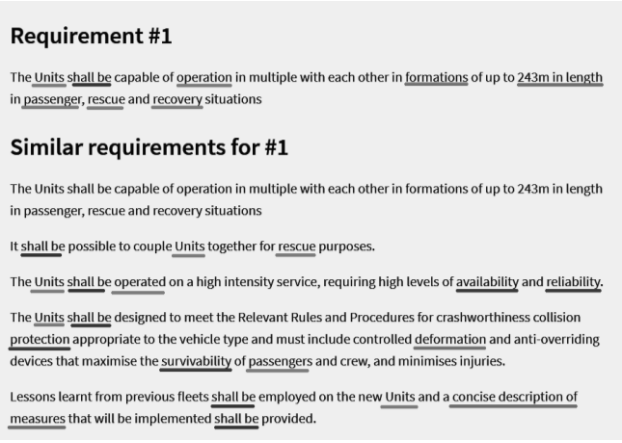


Figure 2. Results of searching for top 5 most similar requirements to a given query requirement in the NLP4RE Prototype: common and similar keywords are highlighted with underlines in the searched and retrieved requirements

To implement this capability, we encode (i.e., vectorize) requirements text semantically and use the distances between the encoded vectors as a similarity measure. So far, we have explored pre-trained language models such as BERT [15], XLNet [16], and T5 [17].

The current prototype works on a local development machine, and it has been empirically tested on the dataset of requirements from 5 projects provided by our industrial partner. Current evaluation results show a high similarity between requirements across the projects (cf. Figure 3) and within the same project (cf. Figure 4). This is due to the fact that in every project, there are typical types of requirements repetitively mentioned about each unit and component of the railway systems, regardless of the market and exact client. The results also show that semantic similarity correlates with other features, such as team allocation, to a large extent.

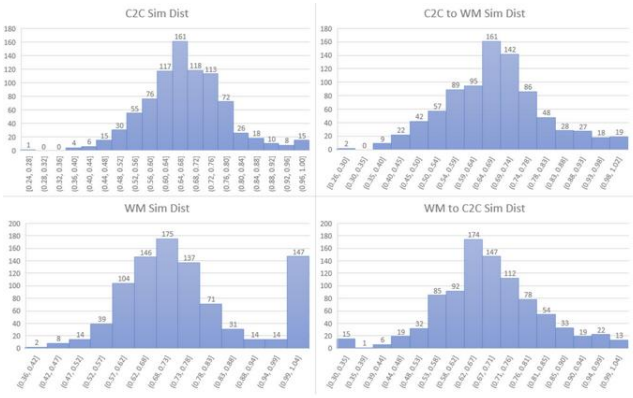


Figure 3. High similarity rate identified between the requirements across different projects

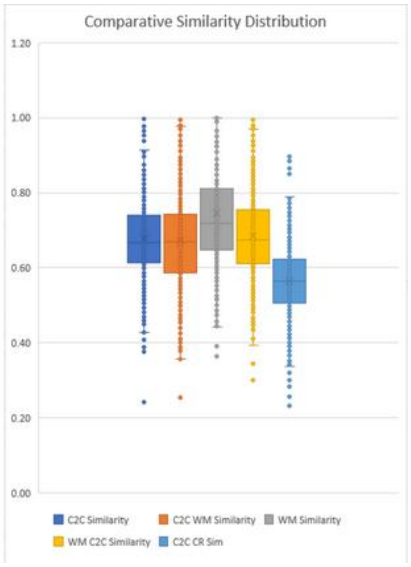


Figure 4. High similarity rate identified between the requirements within each of the 5 projects.

These results still require further validation by domain experts. In fact, the solution will be further evaluated by comparing the actions recommended by the AI system with the actions recommended by a requirement engineer adding the possibility to include automated process steps in the requirement engineering processes.

Conclusions

Nowadays, Natural Language Processing does not supply the similarity check techniques sufficiently advanced for railway industry needs. Therefore, AIDOaRt aims to go beyond the state-of-the-art to introduce them. Such capabilities are going to increase the satisfaction of requirement engineers and end-users as they reduce effort and costs related to their activity.

Our next steps are to validate the results (e.g., with human feedback or based on the similarity of team allocation), to investigate the effect of some data preprocessing techniques and larger language models, and to explore other semantic similarity check approaches. Validation in the next months will consider the function of the AI system will first be qualified by verifying that the actions recommended by the system match those recommended by an expert bid/advance engineer. We would then collect and integrate experts' feedback and add more information to output: requirements categories/types, team allocation and compliance. Also, we would like to explore other semantic search approaches, NLP techniques and text similarity measures and to enhance input/output UI and UX. The ambition will also be to apply the studied semantic search in large models (UML, SysML, Archimate, TOGAF) to easily find similar model elements, for instance requirements, given their description.

Acknowledgements

Special thanks to all AIDOaRt consortium members who are currently working on this project. The AIDOaRt project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350.The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, and Spain.

References

[1] AIDOaRT website: Available at: https://www.aidoart.eu/ (Accessed: 09-04-2024)

[2] Modelio website: Available at: www.modelio.org (Accessed: 09-04-2024)

[3] Modelio Open Source, Available at: https://github.com/ModelioOpenSource/Modelio (Accessed: 09-04-2024)

[4] Hackathon AI-Augmented Requirements Engineering for the Railway Industry , Available at: https://www.linkedin.com/pulse/ai-augmented-requirements-engineering-railway-industry-aidoart/ (Accessed: 09-04-2024)

[5] Hackathon Achieving Explainable AI to Support Decision-making during Requirement Engineering Analysis, Available at: https://www.linkedin.com/pulse/achieving-explainable-ai-support-decision-making-during-requirement-ta7hf/ (Accessed: 09-04-2024)

[6] V. Ivanov, A. Sadovykh, A. Naumchev, A. Bagnato, K. Yakovlev, Extracting Software Requirements from Unstructured Documents, AIST 2021: 17-29, 2021.

[7] H. Bruneliere, V. Muttillo, R. Eramo, L. Berardinelli, A. Gómez, A. Bagnato, A. Sadovykh, A. Cicchetti. AIDOaRt: AI-augmented Automation for DevOps, a model-based framework for continuous development in Cyber-Physical Systems, Microprocess. Microsystems 94: 104672 (2022)

[8] AIDOaRT Alstom Use case website: Available at: https://www.aidoart.eu/aidoart/use-cases/3 (Accessed: 09-04-2024)

[9] A. Bajceta, M. Leon, W. Afzal, P. Lindberg, M. Bohlin, "Using NLP tools to detect ambiguities in system requirements - A comparison study," CEUR Workshop Proceedings, vol. 3122, 2022.

[10] L. Zhao et al., "Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study." arXiv, Apr. 07, 2020. doi: 10.48550/arXiv.2004.01099.

[11] F. Nazir, W. H. Butt, M. W. Anwar, and M. A. K. Khattak, "The applications of natural language processing (NLP) for software requirement engineering-a systematic literature review," in International conference on information science and applications, Springer, 2017, pp. 485–493.

[12] M. Abbas, A. Ferrari, A. Shatnawi, E. Enoiu, M. Saadatmand, and D. Sundmark, "On the relationship between similar requirements and similar software: A case study in the railway domain," Requirements Eng, Jan. 2022, doi: 10.1007/s00766-021-00370-4.

[13] Sadovykh, A., Yakovlev, K., Naumchev, A., Ivanov, V., Natural Language Processing with Machine Learning for Security Requirements Analysis: Practical Approaches, CyberSecurity in a DevOps Environment. Springer, Cham. (https://doi.org/10.1007/978-3-031-42212-6_2), 2014.

[14] D. Chandrasekaran and V. Mago, "Evolution of Semantic Similarity -- A Survey," ACM Comput. Surv., vol. 54, no. 2, pp. 1–37, Mar. 2022.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, Minneapolis, pp. 4171–4186, Jun. 2019.

[16] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding." arXiv, Jan. 02, 2020. doi: 10.48550/arXiv.1906.08237.

[17] C. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." arXiv, Sep. 19, 2023. doi: 10.48550/arXiv.1910.10683.

5G-Enabled Edge Computing for Real-Time Smart Mobility Applications: The PROXIMITY Platform

Elli Kartsakli, Oriol Martínez, Imanol Rojas, Alba Cañete, Vicente Masip, Eduardo Quiñones
Barcelona Supercomputing Center; email: elli.kartsakli@bsc.es

Abstract

Edge computing is an emerging paradigm that addresses the need for real-time processing close to the data sources, which becomes imperative as the volume of data generated by the massive and dispersed Internet of Things (IoT) devices grows exponentially. At the same time, the fifth generation of communications (5G) strives to deliver ubiquitous, fast and reliable connectivity, enabling a wide range of vertical services in sectors such as smart cities, autonomous driving, healthcare, and industrial automation, by supporting massive device connectivity and ultra-low latency communication. The convergence of both technologies can provide efficient, scalable, and responsive services that meet the demands of a rapidly evolving digital landscape.

This paper presents the approach proposed in the PROXIMITY project to provide a unified, integrated and 5G-enabled communication ecosystem, aiming to facilitate the development, deployment and execution of innovative services than can run anywhere across the edge and cloud compute continuum. To showcase the capabilities of this approach, two smart mobility use cases have been selected, implementing real-time analytic services for safe and clean mobility.

Keywords: edge computing, compute continuum, 5G, orchestration, smart mobility.

1 Introduction

The path to innovation is paved by the recent advances in the Information and Communication Technology (ICT) sector, spanning across several interconnected domains that have been evolving in parallel. The rise of the *Internet of Things (IoT)* has created an ecosystem of sensing devices, which generate a huge volume of data that can be converted to valuable knowledge and actionable insights, leveraging big data analytics and *Artificial Intelligence (AI)*. In the last decade, cloud computing has been a key enabler for these technologies, providing the required processing power and storage capacity. However, as the number of connected and smart devices (IoT sensors, smartphones, connected vehicles, etc.) grows exponentially, cloud-based centralised solutions do not scale anymore and the transfer and processing time of the huge volumes of data to remote cloud servers is no longer efficient, introducing latencies that cannot be tolerated by applications with real-time requirements.

The need for scalable solutions and ultra-fast response times can be met by bringing the data analytics and AI methods closer to the data sources, a concept known as *edge computing*. Moreover, the advent of the *fifth generation of communications (5G)* is further driving the adoption of edge computing, bringing significant performance enhancements in terms of bandwidth, ultra-low latency, and massive device connectivity, along with a high level of network programmability.

Despite the clear potential of the aforementioned technologies, it is the design of real-world mobility applications addressing the specific needs of smart and clean mobility that will generate a tangible value for the European economy, and benefit the involved stakeholders, end users and society in general. Unfortunately, this innovation is hindered by the lack of integrated and unified computing and communication frameworks capable of developing, deploying and efficiently executing smart mobility applications.

This paper presents the approach proposed in the PROXIMITY project. PROXIMITY provides a software framework for the development, deployment and execution of innovative data analytics applications over a unified, integrated and multi-tenant computation and 5G-enabled communication ecosystem. Section 2 will present the key challenges motivating this work, whereas Section 3 will introduce the PROXIMITY concept and implemented platform. The two use cases that are being developed to showcase the potential of the proposed solution will be described in Section 4, followed by conclusions and next steps in Section 5.

2 Challenges

The concept of PROXIMITY has been motivated by two specific technical challenges:

- *Challenge 1 – Handling the heterogeneity and multi-tenancy of the compute computing environment:* The highly heterogeneous nature of edge and cloud infrastructures, which vary significantly in computing power, processor architectures, and networking capabilities, introduces considerable complexity to application development and deployment. Furthermore, the geographic dispersion of IoT data sources, for example within a smart city, drives the need for a distributed approach to computing and storage. Finally, the shared nature

of computing (especially at the edge) and network infrastructure resources among multiple tenants requires advanced orchestration and lightweight virtualization mechanisms, such as containers and serverless platforms, that can meet the specific needs of each application. This is particularly important for latency-sensitive real-time services, where performance reliability and predictability must be maintained.

- *Challenge 2 – Creating a unified communication of the compute continuum for the ubiquitous distribution and execution of AI and complex big data analytics:* IoT deployments involve a mix of communication technologies such as WiFi, fiber optics, and Vehicle-to-Everything (V2X) whern connected vehicles ae involved, that are not fully integrated into 5G, resulting in a heterogeneous communication environment. On the other hand, the edge computing landscape is not consolidated. The European Telecommunications Standards Institute (ETSI), through the Multi-access Edge Computing (MEC) Industry Specification Group (ISG), is developing a reference architecture to bring cloud-computing capabilities to the edge of the network, specifying the elements needed to enable applications to be hosted on multi-vendor multi-access edge computing environments [1]. At 3GPP, the edge computing revolves around the definition of MEC with focus on 5G [2], whereas both initiatives are collaborating in an effort to align their approaches [3]. However, there are still open challenges such as the design of interfaces to expose network services to applications or the implementation of local breakouts to deliver ultra-low latency services by keeping data at the edge.

3 Proximity

To address the aforementioned challenges, PROXIMITY is designing a software framework for the development, deployment and execution of innovative data analytics applications over a unified, integrated and multi-tenant computation communication ecosystem. This design combines multi-disciplinary research in the cloud and distributed computing, software development and telecommunications, aiming to: i) abstract the logic from the underlying infrastructure by employing appropriate programming models, ii) unify edge computing with the 5G communication environment, and iii) design orchestration mechanisms taking into account real time monitoring information of the platform and deployed services.

PROXIMITY leverages the BSC COMP Superscalar (COMPSs) framework [4] for the development and deployment of complex analytics workflows. COMPSs offers a task-based programming model based on sequential development in which the user simply annotates the functions to be executed as asynchronous parallel tasks and the data dependencies among them. The runtime system is then in charge of exploiting the inherent concurrency of the code, automatically enforcing the data dependencies between tasks and spawning these tasks to the available resources. To deal with the heterogeneous compute continuum environment (*Challenge 1*), PROXIMITY employs a system model that combines a

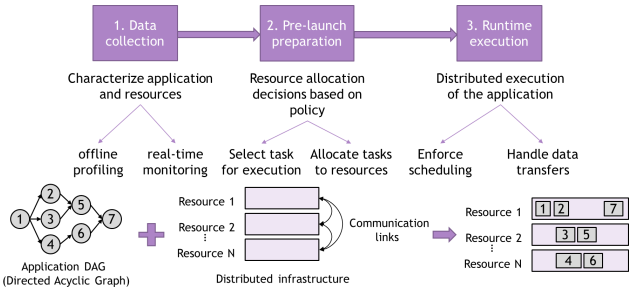


Figure 1: The task-based scheduling PROXIMITY orchestrator

Direct Acyclic Graph (DAG) representation to capture the dependencies among the tasks with a Digraph (directed graph) compute continuum model to characterize the heterogeneous computing resources and the respective communication links.

Figure 1 provides an overview of the task-based scheduling process, consisting of three phases. At the first phase, the characterization of the application and compute continuum resources takes place. This characterization has been initially based on extensive profiling of both the application and the compute continuum to estimate the execution time upper bound of each task and the transfer time of any data dependencies. PROXIMITY further implements a monitoring mechanism to extract this information during runtime, making it possible to capture any changes in the compute continuum and enable dynamic orchestration decisions. Then, the scheduling decisions are made based on the implemented policy. This decision includes the task prioritization (e.g., based on factors such as the task criticality or the number of successors) and the mapping of tasks to the available computing resources. Finally, during runtime, the scheduling decisions are enforced, together with the necessary data transfers honoring the task dependencies.

This software architecture has been deployed over the PROXIMITY platform, depicted in Figure 2. To enable a unified 5G-enabled compute continuum (*Challenge 2*), PROXIMITY has deployed a cloud-native 5G software stack, making the necessary adaptations to support data forwarding and processing at the edge. Open5GS has been used as the open source platform implementing 5G core network functions of the 3GPP release 17 and following the 5G Service Based Architecture (SBA) principle. In PROXIMITY, Open5GS has been configured in a 5G Standalone Architecture (SA). The core has two main planes: the control plane and the user plane, which are physically separated as CUPS (control/ user plane separation) is implemented. The 5G SA control plane functions are configured to register with the NF Repository Function (NRF), which then helps them discover the other core functions. The Access and Mobility Management Function (AMF) handles connection and mobility management and is the connecting point with the RAN network (gNB). Session management is all handled by the Session Management Function (SNF). The NSSF (Network Slice Selection Function) provides a way to select the network slice, and

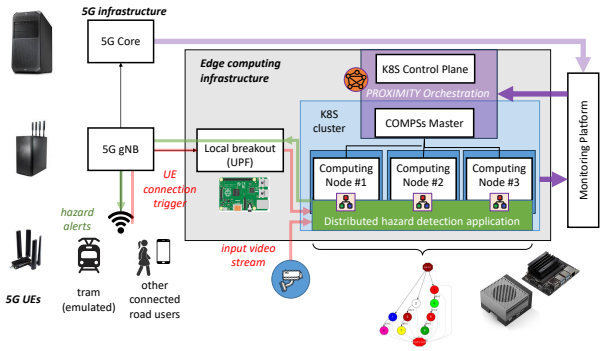


Figure 2: The PROXIMITY 5G-enabled edge computing platform

Policy and Charging Function (PCF) is used for charging and enforcing subscriber policies, Other core functions are also available, to implement other control plane functionalities regarding authentication, subscription policies, secure roaming, etc. The 5G SA core user plane is much simpler, as it only contains a single function, namely the User Plane Function (UPF).

At the Radio Access Network (RAN) the Amarisoft callbox classic [5] has been used as the 5G base station (gNB). The gNB is responsible for handling the radio communication with the User Equipment (UE), including providing radio access, mobility management, and the establishment of data sessions. For the UEs, two different equipment have been used with programmable Sysmocom SIM cards: i) a commercial 5G smartphone (Samsung galaxy A23), ii) a 5G USB dongle [6] equipped with a 5G-modem (Quectel RM502Q-AE [7]) and connected to a computing device (e.g., a laptop, etc.).

In a traditional 5G network, data from the UEs is sent to the gNB (acting as the radio access point), and then forwarded to the core network UPF for routing. From the UPF, the data is directed to the appropriate external network (e.g., the internet or a private network, depending on the service). To enable edge computing, a second instance of the UPF has been deployed at the edge infrastructure, on a dedicated Raspberry Pi node. This function serves as the local breakout, enabling user data to be forwarded for processing to the edge cluster, thus reducing latency by handling data closer to the data sources.

The edge cluster of the PROXIMITY platform consists of heterogeneous GPU-enabled nodes from the Nvidia Jetson family (i.e., Nvidia Jetson, AGX Orin and Orin NX). Kubernetes (K8S) [8] has been employed as the underlying resource and container orchestration technology. K8S provides the necessary abstraction to automate the deployment, scaling, and management of cloud-native containerized services on top of the heterogeneous compute continuum infrastructure. The K8S master is deployed in one of the available computing nodes, whereas the rest of the available resources for the K8S nodes of the cluster (marked as blue boxes in Figure 2), where computation may take place. In each node, the services may run as containers within the K8S pods. The K8S scheduler allows applications to adjust resource usage dynamically by adding or removing pods based on demand. It supports both horizontal scaling (increasing/decreasing the number of pods) and vertical scaling (adjusting the resources like CPU and memory of running pods) to ensure efficient resource utilization and high availability.

The BSC COMPSs framework has been employed as the runtime environment to deploy, execute and orchestrate distributed workflows. COMPSs has a master-worker structure. By employing the COMPSs programming model, sequential applications can be transformed to distributed COMPSs tasks, which can then be deployed and executed across the COMPSs workers. COMPSs has its own scheduler which provides fine-grained orchestration of the distributed tasks, i.e., selects the order in which tasks can be executed and maps them on the most appropriate resources to meet the required performance (e.g., in terms of latency). In PROXIMITY, COMPSs has been extended to support deployment of the master and workers processes as K8S services. The integration between the two technologies enables different scales of *orchestration* decisions, combining the K8S resource scaling capability with the COMPSs task-based scheduling, to allocate and prioritize the application tasks while leveraging parallel execution.

Finally, the PROXIMITY monitoring layer has been implemented based on Prometheus [9], which is an open-source monitoring and alerting toolkit designed for reliability and scalability, primarily used to collect and query metrics from various services and systems. Prometheus integrates seamlessly with K8S to provide powerful monitoring and alerting by scraping metrics from the K8S API, nodes, and pods, enabling detailed insights into cluster performance and health. COMPSs applications have also been adapted to report runtime metrics (such as the task completion time and the end-to-end worfklow execution time) to Prometheus. Finally, the necessary endpoints are also being developed to enable the collection of 5G-related metrics, thus creating a unified observability stack for the platform.

4 PROXIMITY Use Cases

Two smart mobility use cases have been selected to showcase the capabilities of PROXIMITY solution to deliver real-time analytic services leveraging edge computing and 5G. The first use case implements a hazard detection application which aims to improve road safety by analyzing and promptly detecting potential risk situations in urban traffic scenarios. The second use case builds upon the information collected by the first application on the real time traffic, in order to extract estimations on the air quality. The execution of both applications over the same infrastructure as a unified workflow provides a representative example on how the PROXIMITY platform can handle multi-tenancy, ensuring that the performance of both applications meets the required Quality of Service (Qos).

4.1 Use case 1: Real-time hazard detection

The key objective of the real-time hazard detection application is the timely identification of hazardous situations that may lead to traffic accidents in urban areas, especially involving the interaction of the tramway network with other public or private transportation and the generation of warnings to

involved connected parties. Secondary objectives include the detection of traffic statistics (congestions, traffic violations, etc.) that can be used to design more effective traffic management policies. Overall, this use case will address the need for smarter, safer and connected mobility. The hazard detection application takes as input the live video stream of traffic cameras and performs the following functionalities: 1) object detection, tracking, estimation of dynamic properties (e.g., position, speed, orientation); 2) data aggregation and deduplication of data from multiple cameras partially covering the same area; 3 trajectory prediction for moving vehicles; and 4) hazardous detection and generation of corresponding alerts. The PROXIMITY framework will enable the development of this application as a single data analytics workflow composed of different data analytics methods, which will be distributed across the compute continuum, from edge to cloud, exploiting the inherent parallelism of the application. The PROXIMITY orchestrator will handle the placement and scheduling of methods on-the-fly, based on the computing and communication infrastructure constraints (e.g., processing power, link quality, etc.), the latency requirements of each task and the end-to-end application execution time. Edge computing will be leveraged to process and deliver the generated alerts to the connected end users (e.g., the tram driver) with ultra-low latency.

4.2 Use case 2: Vehicular emissions and air quality estimation

Leveraging the infrastructure and object detection analytics methods of the first use case, the second PROXIMITY use case addresses the need for clean transport and mobility, offering the means to understand and monitor the environmental impact of urban transportation and driving behavior. To that end, two contributions are envisioned: 1) to provide a fine-grained microscopic model for the estimation of vehicle-related emissions based on real-time image processing, and 2) employ this real-traffic pollution estimation to improve the accuracy of existing models for the prediction of the air quality. Air quality sensors measure pollutant concentrations, without distinguishing between vehicle-related emissions and other sources of contamination (from buildings, factories, etc.). Moreover, such measurements are affected by meteorological conditions, further complicating the task of isolating the impact of traffic. Hence, vehicle-related pollution is mainly obtained by emission models that provide macroscopic estimations (e.g., per year) based on long-term statistics on the vehicle population at a given city. Few solutions exist for microscopic estimation (e.g., per hour), typically relying on traffic simulation models adapted to the area under study.

In this context, the PROXIMITY framework aims to enhance the existing microscopic estimation solutions by using real traffic measurements regarding the type and dynamics (speed and acceleration) of vehicles detected by the real-time video processing analytics. This will enable the estimation of several pollutants at a very fine granularity of few minutes (or even seconds), and so capturing the impact of specific traffic incidents and driving patterns (e.g., traffic congestion, trucks waiting with engine on, etc.). Moreover, this information will allow enhancing the CALIOPE-Urban air quality model [10].

5 Conclusions

This paper has presented the concept and implementation of the PROXIMITY platform, aiming to tackle the challenges that arise on the efficient deployment and execution of real-time services over a highly heterogeneous compute continuum, enabled by 5G communications. The key software components have been deployed over a 5G-enabled edge computing testbed designed to meet the requirements of this project. In this context, two use cases have also been selected to showcase the capabilities of the proposed solution. The next phase of PROXIMITY will focus on the implementation and deployment of the two use case applications on the testbed and the refinement and thorough performance evaluation of the PROXIMITY orchestration solutions to meet the specific application requirements.

Acknowledgements

The PROXIMITY project is being funded by PID2021-124122OA-I00/AEI/10.13039/501100011033/FEDER, UE.

References

[1] ETSI GS MEC 003 V3.2.1 (2024-04), “Multi-access Edge Computing (MEC); Framework and Reference Architecture.”

[2] ETSI TS 123 548 V18.7.0:2024-10), “5G; 5G System Enhancements for Edge Computing; Stage 2 (3GPP TS 23.548 version 18.7.0 Release 18).”

[3] ETSI TR 128 903 V18.0.1 (2024-05)), “5G;Study on alignment with ETSI MEC for edge computing management (3GPP TR 28.903 version 18.0.1 Release 18).”

[4] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia, “Servicess: An interoperable programming framework for the cloud,” *J. Grid Comput.*, vol. 12, p. 67–91, Mar. 2014.

[5] Amarisoft: Amari callbox classic, “https://www.amarisoft.com/test-and-measurement/device-testing/device-products/amari-callbox-classic.”

[6] Waveshare 5G dongle, “https://www.waveshare.com/usb-to-m.2-b-key.htm.”

[7] Quectel 5G modem, “https://www.quectel.com/product/5g-rm502q-ae/.”

[8] Kubernetes, “https://kubernetes.io/.”

[9] Prometheus monitoring solution, “https://prometheus.io/.”

[10] J. Benavides, M. G. Snyder, M. Guevara, A. Soret, C. P. García-Pando, F. Amato, X. Querol, and O. Jorba, “Caliope-urban v1.0: coupling r-line with a mesoscale air quality modelling system for urban air quality forecasts over barcelona city (spain),” *Geoscientific Model Development*, 2019.

Towards Model-Based System Engineering for Cyber-physical Systems in the MYRTUS Project

Alessandra Bagnato, Juan Cadavid

Softeam, Softeam Software, Docaposte Groupe, 3 avenue du Centre 78280 Guyancourt.; Tel: +33 1 30 12 18 58; e-mail: Alessandra.bagnato@docaposte.fr

Abstract

The MYRTUS[1] project aims at unlocking the new living dimension of Cyber Physical Systems (CPS) integrating edge, fog and cloud computing platforms. This integration requires the reinvention of programming languages and tools to orchestrate collaborative distributed and decentralised components. Additionally, components must be augmented with interface contracts covering both functional and non-functional properties. This paper describes the Model-based approach that will be used during the project, including the key cloud standard, the TOSCA (Topology and Orchestration Specification for Cloud Applications) to be used to describe cloud computing services and their components, as well as the orchestration process needed to manage them.

Keywords: *Cyber Physical Systems, CPS,* Requirements Engineering, AI, model-driven engineering.

1 Introduction

Cloud, Edge, and IoT Continuum (CEI) are changing drastically our world, providing a very fertile ground for new challenges and new approaches for dependable and Cyber-Physical System (CPS) Engineering.

The MYRTUS Project[1], within the European Cognitive Computing Continuum cluster [2] , contributes to enhance openness and strategic autonomy in the evolving data and AI economies, while nurturing European value chains and accelerating the digital and green transitions. By establishing adaptive hybrid computing, cognitive clouds, and edge intelligence, it contributes to laying the groundwork for strategic industrial cooperation and building open platforms critical to European competitiveness [9].

MYRTUS [13] embraces the sustainable and responsible computing paradigm, promoting obsolescence avoidance (supported by MYRTUS principle of openness, interoperability, and portability) and resource saving and energy efficiency (supported by HW specialisation and optimization techniques) uniting under a unique modelling environment, Modelio, the whole model side of its design and programming environment.

The paper analyses the first steps towards the Model-based System Engineering for Cyber Physical Systems approach to be used in the MYRTUS project.

The following sections describe the MYRTUS project's objectives and consortium, the Modelio modelling environment and its required modules and the modelling context in MYRTUS.

2 The MYRTUS Project

Funded by the European Union MYRTUS [1] is a Horizon Europe project involving 16 organizations belonging to both the academic and the industrial world, grouped in clusters from 7 different countries, focusing on in Cyber-Physical Systems (CPS). MYRTUS began on 1st January 2024 for a period of 36 months.

On the academic side, it includes:

1. Universita degli Studi di Sassari
2. Technische Universitaet Dresden
3. Universita' degli Studi di Cagliari
4. Lakeside Labs
5. Universidad Politecnica de Madrid
6. King's College London
7. Universita' della Svizzera Italiana
8. Canon Research Center
9. TNO

From the industrial side, it includes:

1. Abinsula Srl
2. Softeam
3. Hiro Microdatacentres
4. Forge Reply Srl
5. ArubaKube Srl

The MYRTUS project aims at unlocking the new living dimension of CPS, embracing the principles of the EUCloudEdgeIOT [2] Initiative, aiming to seamlessly integrate edge, fog, and cloud computing platforms. This integration requires the reinvention of programming languages and tools to orchestrate collaborative distributed

[1] https://myrtus-project.eu/

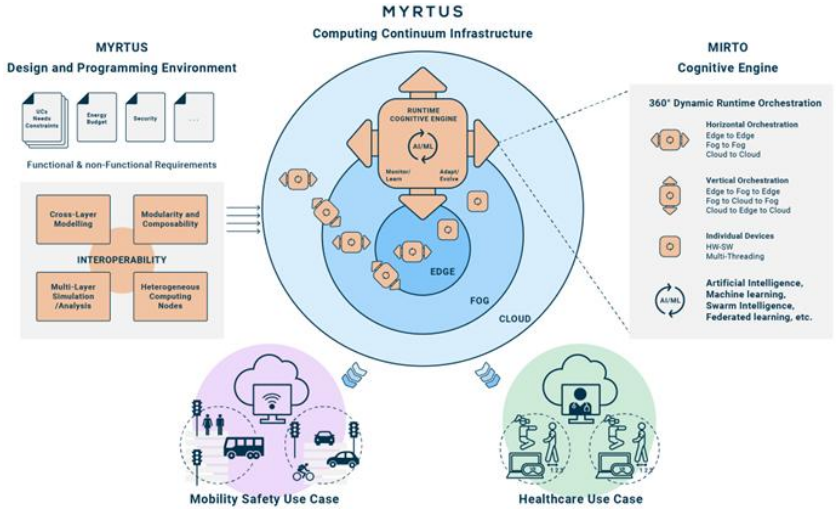


Figure 1: Overview of MYRTUS approach

band decentralised components. Additionally, components must be augmented with interface contracts covering both functional and non-functional properties. MYRTUS solutions play a crucial role in enabling sustainable computing and trustworthiness in CPS.

A key aspect of MYRTUS is its emphasis on runtime adaptivity across all infrastructure layers. The project introduces the Multi-layer 360° Dynamic RunTime Orchestration (MIRTO) Cognitive Engine, a distributed system responsible for orchestrating a scalable, dynamic, and heterogeneous infrastructure. MIRTO leverages self-awareness through feedback control loops to implement self-reconfiguration capabilities, including self-healing and self-optimization. This engine continuously monitors resources and performs runtime orchestration to guarantee high performance and energy efficiency, while upholding security and trust. The project also explores the extensive use of federated learning, swarm intelligence, and distributed strategies to optimize workload distribution and effectively utilize the computing continuum [9].

The following elements are appearing in Figure 1 and are part of the solution [5]:

- The **MYRTUS Reference Infrastructure** that fosters the convergence of diverse fog-level and edge-level devices with cloud computing resources. This integrated architecture forms a computing continuum specifically designed to address the computational demands of complex and dynamic systems, particularly those encompassing Cyber-Physical Systems (CPS) with living characteristics.
- The **MYRTUS Runtime Orchestration Scheme** with a 360° dynamic approach. This orchestration is realized through the MIRTO AI-powered cognitive engine. MIRTO ensures high performance and energy efficiency while upholding critical security and trust requirements.
- The **MYRTUS Design and Programming Environment** (DPE). This DPE serves as a reference design and programming environment specifically tailored for continuum computing systems. The MYRTUS DPE facilitates the development process through interoperable support for various functionalities: cross-layer modeling, threat analysis, design space exploration, application modeling, component synthesis, and code generation.

3 Modelio

Softeam is a technology provider that provides consulting training and engineering services. It counts about 1400 employees, based in Paris, with subsidiaries in Saint Quentin en Yvelines, Rennes, Nantes, Toulouse and Sophia Antipolis. It publishes modelling software for design, development, and automation of application production. One of the Softeam Software department main activities is the development and support of the Modelio CASE Tool, the next generation of the 'Objecteering' CASE tool, which has been on the market since 1992. From 2019 Softeam has become an autonomous subsidiary of DocaPoste, the IT branch of French Post Service Company ("La Poste") to join a team developing innovative solutions in AI, blockchain and e-health.

Modelio is first and foremost a modeling environment, supporting a wide range of models and diagrams and providing many services facilitating the modeling of architectures, such as model consistency-checking.

Within MYRTUS Softeam will extend: the threat modelling module in the Attack Tree (AT) Designer Modelio Module [6], and Modelio with import/export capabilities of TOSCA [8][9] models.

4 The TOSCA standard

The TOSCA (Topology and Orchestration Specification for Cloud Applications) [8] is a standard by OASIS (Organization for the Advancement of Structured Information Standards) [12], an international nonprofit consortium that promotes open, collaborative development of e-business specifications based on public standards such as XML and SGML.

TOSCA is used to describe cloud computing services and their components, as well as the orchestration process needed to manage them. It enables the standardization of cloud-based services, simplifies application deployment to any cloud platform, and supports multi-cloud environments. TOSCA is an open standard supported by various cloud-related platforms and orchestration tools, and it is highly extensible, allowing developers to add vendor- or domain-specific mechanisms.

Originally conceived for cloud services and applications, it is being extended to IoT/Edge/Fog by TOSCA TC through an ad-hoc Workgroup exploring these new domains.

The TOSCA language describes cloud services using templates and plans. Templates define the structure of a cloud service.

Plans define the processes that start, stop and manage that cloud service over its lifetime. For example, TOSCA could be used to describe the relationship between Docker containers, virtual machines, server components, endpoints and services within a cloud environment [8][9]. This enables faster, repeatable and scalable application deployments.

5 The Attack Tree Designer

The Attack Tree Designer [6] is an Open-Source Modelio module conceived for designing Attack Trees . Attack Tree Designer is deployed as a module in Modelio modelling environment.

The Attack Tree Designer module allows users to design attack trees diagrams showing how an asset or a target might be attacked. These diagrams are intended for security specialists for modelling the attacks that occur on IT systems or cyber-physical systems and describe the events that lead to the attack in the form of a tree in which the attack is represented by the root element which is related with “OR” and “AND” conditions to the children that represent the sequence of events that lead to the root attack.

This module offers an ergonomic environment for designers and contains additional set of features that allow users to configure security attributes for the attacks such as the severity and the likelihood of the attack. It allows users to attach countermeasures to their attacks and to detect encountered attacks. Moreover, it allows referencing other trees, importing and exporting attack trees.

Figure 2 show the Attack Tree (AT) “Impersonate Operator”.

6 Modeling in MYRTUS

The model-side of the MYRTUS design and programming environment (the MYRTUS DPE) will be based on Modelio.

The functional partitioning of the overall use case scenarios is defined starting from the high-level UML-based functional specification exploiting the activity [10] and TOSCA-compliant [8] component diagrams in Modelio. If the TOSCA infrastructure description is not available, then generic component diagrams are used. UML (Unified

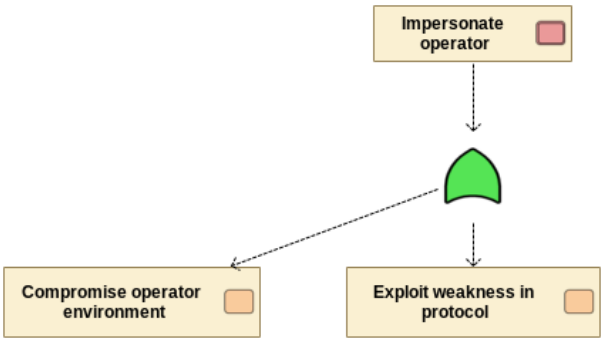


Figure 2: Attack Tree “Impersonate Operator”

Modeling Language) is a modeling language standardized by the OMG [11].

Softeam will implement a TOSCA Designer, an open-source Modelio extension to specify multiple aspects/domains related to multi-/cross-cloud applications. The TOSCA’s open-source domain specific language [7] providing model-based descriptions of services, platforms, infrastructure, and data components, along with their relationships, requirements, capabilities, configurations, and operational policies will be used.

The MYRTUS design stage incorporates a comprehensive security threat analysis process. This analysis leverages Attack Trees (ATs) to systematically identify potential attack scenarios for a given threat. Security and privacy requirements are treated with the same importance as other system requirements and are modeled using Modelio as Key Performance Indicators (KPIs) with quantifiable metrics for assessment. Additionally, the project introduces ADT, an extension of ATs that incorporates defense measures. ADT models will be employed to represent the interaction between potential attackers and the implemented security measures, enabling the evaluation of mitigation techniques for each identified threat.

In MYRTUS, the Modelio ADT Extension will automatically synthesise the code to mitigate the security threats starting from the Attack Defence Tree, e.g. a security requirement could impose the signature of firmware to ensure trustworthiness. The attack tree will be used to model the potential modification of firmware. The mitigation technique, the signature of the firmware, will be modelled in a similar way.

Figure 3 shows the Attack Tree (AT) “Modify Firmware”.

Softeam will provide support for the usage/extension of Modelio for automatic synthesis of code starting from UML descriptions and will cooperate with the Universita’ della Svizzera Italiana (USI) to define strategies to automatically derive threats countermeasures, selected from a library of countermeasures.

MYRTUS DPE will support security by design, extending Modelio to model novel and not yet explored threats typical of the computing continuum and using them as starting point for mitigating the threats.

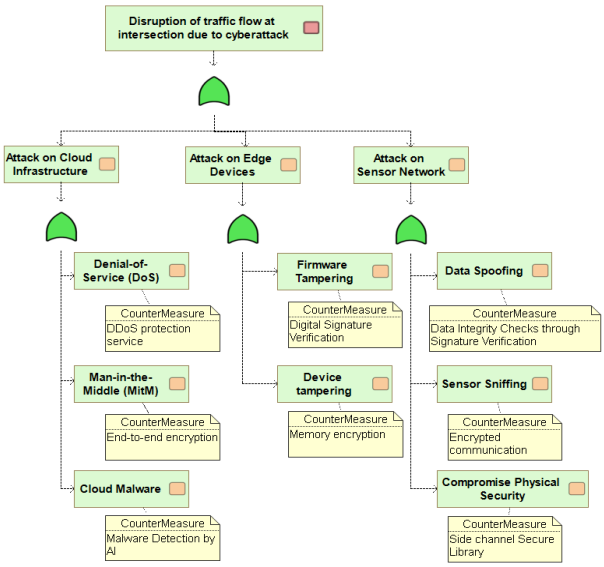


Figure 3: Attack Tree "Modify Firmware"

Modelio models will support decentralized structures and reflect the dynamic changes that involve the agents and computing environment. Additionally, Modelio [3][4] will synthesize countermeasure from those models and will model threats mitigation techniques and properties/functionalities of privacy preserving libraries and components to ensure trust, enhancing the State of the Art in the domain by providing a comprehensive threat-countermeasure strategy.

MIRTO will use the novel models to constantly assess at run time if the privacy, trust, and security requirements are guaranteed ultimately enabling the deployment and widespread adoption of privacy preserving and trust techniques in large and dynamic environments.

7 Conclusions

Nowadays, Model Based Design of Cyber Physical System needs to adapt to the current Cloud, Edge, and IoT Continuum (CEI) world. MYRTUS aims to go beyond the state-of-the-art to introduce and work with several standards, UML (Unified Modeling Language) the modeling language standardized by the OMG, TOSCA, the OASIS standard language widely adopted in the industry for describing the relationships and dependencies between services and applications on cloud computing platforms and the Attack Defence Tree Models to analyse the possible threats to which the Cyber Physical System is exposed to model novel and not yet explored threats typical of the computing continuum and using them as starting point for mitigating the threats.

Acknowledgements

Special thanks to all MYRTUS consortium members who are currently working on this project. MYRTUS is funded by the European Union, by grant No. 101135183. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] MYRTUS website: Available at: https://myrtus-project.eu/ (Accessed: 09-04-2024)

[2] EUCloudEdgeIoT.eu initiative website: Available at: website: Available at: https://eucloudedgeiot.eu/(Accessed: 09-04-2024)

[3] Modelio website: Available at: www.modelio.org (Accessed: 09-04-2024)

[4] Modelio Open Source, Available at: https://github.com/ModelioOpenSource/Modelio (Accessed: 09-04-2024)

[5] T. Fanni, M. K. Zedda, F. Palumbo, "MYRTUS- Multi-layer 360° dYnamic orchestration and interopeRable design environmenT for compute-continUum Systems", Available at: https://zenodo.org/records/11143796 (Accessed: 24-06-2024)

[6] Attack Tree Designer Module, Available at: https://github.com/Modelio-R-D/AttackTreeDesigner (Accessed: 09-04-2024)

[7] TOSCA-Simple-Profile-YAML, Available at: http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csd03/TOSCA-Simple-Profile-YAML-v1.0-csd03.html (Accessed: 09-04-2024)

[8] TOSCA, OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC, Available at: https://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=f9412cf3-297d-4642-8598-018dc7d3f409 (Accessed: 09-04-2024)

[9] Introducing the Cognitive Computing Continuum Cluster Projects: Pioneering the Future of AI and Edge Computing, Available at: https://eucloudedgeiot.eu/introducing-the-cognitive-computing-continuum-cluster-projects-pioneering-the-future-of-ai-and-edge-computing/ (Accessed: 09-04-2024)

[10] Unified Modeling Language™ (UML®). Available at: http://www.omg.org/spec/UML/ (Accessed: 09-04-2024)

[11] Object Management Group® Standards Development Organization (OMG® SDO) http://www.omg.org/(Accessed: 09-04-2024)

[12] OASIS, Organization for the Advancement of Structured Information Standards. Available at: https://www.oasis-open.org/org/(Accessed: 09-04-2024)

[13] F. Palumbo, M. K. Zedda, T. Fanni, A. Bagnato, L. Castello, J. Castrillon, R. Del Ponte, Y. Deng, B. Driessen, M. Fadda, T. Halna du Fretay, J. de Oliveira Filho, V. Rao, F. Regazzoni, A. Rodríguez, M. Schranz, G. Sedda, "MYRTUS: Multi-layer 360° dYnamic orchestration and interopeRable design environmenT for compute-continUum Systems", Invited Paper, *Proceedings of the 21st ACM International Conference on Computing Frontiers Workshops and Special Sessions (CF '24 Companion*), May 7–9, 2024, Ischia, Italy. ACM, New York, NY, USA, 2024. https://doi.org/10.1145/3637543.3654618.

LIONESS – Improving and Leveraging OpenMP for the Efficient and Safe Use of New High-Performance Hardware Platforms

Sara Royuela
Barcelona Supercomputing Center; email: sara.royuela@bsc.es

Franck Wartel, Sylvain Tiberio
Airbus Defence and Space; email: {franck.wartel, sylvain.tiberio}@airbus.com

Eric Jenn
IRT Saint Exupéry; email: eric.jenn@irt-saintexupery.com

Hubert Guérard, Guy Bois
Space Codesign; email: {hubert.guerard, guy.bois}@spacecodesign.com

Abstract

The number and diversity of embedded Field-Programmable Gate Arrays (FPGAs) Multi-Processor Systems On Chip (MPSoCs) in modern satellites is increasing, and so is the complexity and cost of using them efficiently (i.e., optimally exploiting the available resources) and safely (i.e., complying with the applicable safety and availability constraints). Programming languages traditionally used in critical real-time systems have yet to be designed to address the extreme parallelism of modern platforms. To address this limitation, OpenMP, the de-facto standard for exploiting parallelism in shared-memory systems in the HPC domain, is increasingly considered a suitable solution in critical domains. OpenMP implements a comprehensive set of computation models (e.g., data and task parallelism, host and accelerator support), comes with an extensive set of assets (e.g., tools, libraries), and supports a large set of CPU and accelerator devices (e.g., GR740, MPPA, NVIDIA Jetson and Xilinx Ultrascale+). Despite preliminary analysis proving the productivity and efficiency of OpenMP in the space, automotive and railway domains, some challenges must be addressed. This paper introduces LIONESS, a project funded by the European Space Agency (ESA) proposing an advanced OpenMP framework that combines enhancements in the parallel programming model with adapted compiler and runtime systems to provide benefits along two axes: (1) resilience, through providing fault-tolerance techniques, and (2) heterogeneity, through enabling the design space exploration of multiple deployment configurations considering multi-cores and accelerator devices.

Keywords: FPGA, Fault-tolerance, Efficiency, OpenMP, Design Space Exploration.

1 Introduction

The increasing demands of onboard payload data processing in terms of higher data rates and massive parallelism have driven the adoption of FPGA devices in the space domain during the last decade. FPGAs provide reduced weight and board space, increased reliability with fewer solder connections, and increased flexibility, allowing in-flight reconfiguration. Nonetheless, the programmability of these devices is still a hurdle, incurring considerable design and development costs. Two main aspects affect *programmability*: 1) the offloading method that communicates the multi-core with the accelerator and 2) the hardware design space exploration (DSE).

Focusing on the offloading challenge, OpenMP is the de facto standard for parallelising shared-memory applications from the HPC domain. The model has been further tested in space, automotive and railway domains by 1) introducing language extensions to model real-time systems (e.g., deadline, priority, events) [1], 2) providing compiler analysis techniques to ensure the correctness in terms of data-races [2, 3], and 3) defining run-time mechanisms that efficiently orchestrate the parallel execution while ensuring system's predictability and schedulability [4, 5, 6]. OpenMP tasks allow for the natural exploitation of parallel architecture opportunities. On top of that, the offloading mechanism provided with the OpenMP accelerator model enables the deployment of host-centric heterogeneous applications while the OpenMP runtime coordinates the execution in a multi-core platform with one or more accelerators, e.g., an FPGA [7]. The LLVM compilation framework has emerged as a common platform for researchers and vendors providing support, at the language level, for OpenMP on top of C/C++ programs and, at the hardware level, for architectures including SPARC and LEON [8].

Focusing on the DSE challenge, the Tool for OpenMP Annotations to Space design Translation (TOAST) [9] and SpaceStudio [10] propose a design flow that allows for avoiding the

manual re-coding of the application into some low-level hardware description language, such as VHDL or Verilog, while exposing the capability to explore various implementation solutions. Specifically, TOAST allows the extraction and encapsulation of OpenMP target code. SpaceStudio provides DSE and platform-independent implementation capabilities, with support for generating executables for different vendors, including Xilinx and Intel Altera.

LIONESS leverages and extends OpenMP, LLVM, TOAST and SpaceStudio to provide a software framework for developing and deploying aerospace applications on parallel and heterogeneous architectures, addressing *resiliency*, *performance*, *heterogeneity*, and *programmability*. This article describes LIONESS's application context, proposed solution, and planned evaluation.

2 Application context

LIONESS has customized a space use case that allows for configuring different levels of performance and resilience requirements, exposing its software solution's potential benefits. Along the same line, an advanced FPGA MPSoC providing different types of hardware architectures has been selected to test LIONESS's benefits. This section introduces both the use case and the advanced hardware platform.

2.1 Use case: A *Corner Detection* Method

In the space domain, image processing is used in Vision-Based Navigation (VBN) systems to improve spacecraft position estimation and enable processes like landing and space rendezvous, e.g., automatic docking to the International Space Station (ISS) and space debris capture. LIONESS has selected the corner detection processing chain described in Figure 1 as the use case. This application comprises four chained functions: *gradient calculation*, *gradient aggregation*, *dilatation* and *maximum detection*. The two former functions are convolutional neural networks (CNNs), and the two latter are post-processing methods. The algorithm's outcome is a list of pixel positions of detected corners in the input image.

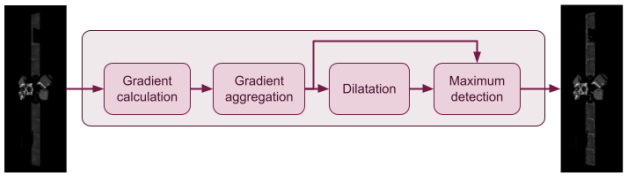


Figure 1: LIONESS use-case: Corner detection pipeline.

The application includes two different configurations:

- *A single image test*: Execute the corner detection on a single image, representing situations like a Lander Hazard Detection and Avoidance (LHDA).
- *Several images test*: Execute the corner detection on several images, representing situations like Feature Tracking (FT).

Both versions execute the same algorithm implementation, but each configuration can be mapped to different functional safety objectives.

2.2 Hardware platform: Xilinx Ultrascale+

LIONESS leverages the hardware capabilities of the Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit [11]. This platform is a general-purpose board that, as exposed in Figure 2, combines the following heterogeneous system:

- A powerful processing system (PS) with the next processing units, among others:
  - Cortex-A53 application processing unit (APU): Arm v8 64-bit quad-core CPU with 8-stage pipelined processor with 2-way superscalar, in-order execution pipeline, separate 32Kb L1 instruction/data caches, 1MB of L2 cache, and a memory management unit (MMU).
  - Cortex-R5 real-time processing unit (RPU): Arm v7 32-bit dual real-time processor, 32KB of instruction and data L1 caches. Each core owns a local 128Kb tightly coupled memory (TCM). The cores can work independently or in a lock-step and support error correction code (ECC) to detect up to two errors and correct any single error.
  - Mali-400 MP2 graphics processing unit (GPU) with pixel and geometry processor and 64 KB L2 cache.
- A user-programmable logic (PL) with 504K LUTs, 38Mb of RAM and 1728 DSP slices.

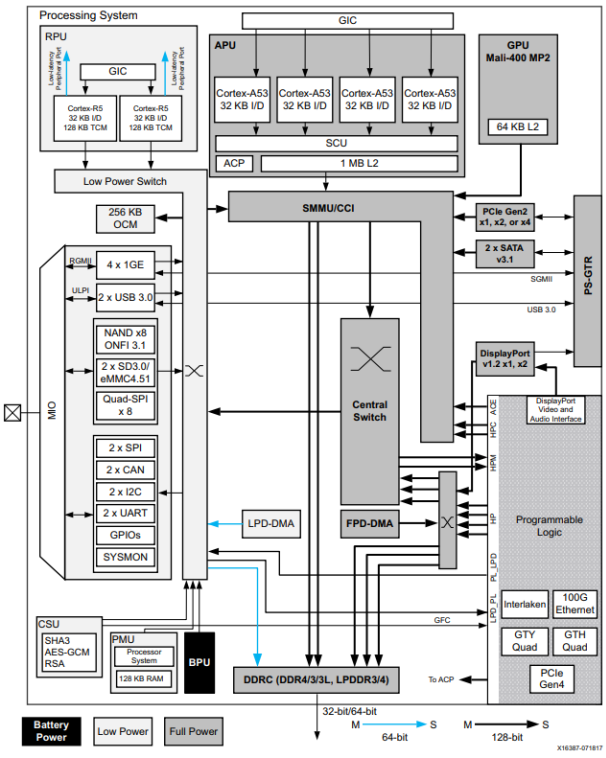


Figure 2: Zynq UltraScale+ ZCU102 top-level block diagram.

The GPU won't be used for the project. Nevertheless, extending LIONESS to work with the GPU would be straightforward since both the OpenMP specification and the LLVM compilation framework support such a heterogeneous system.

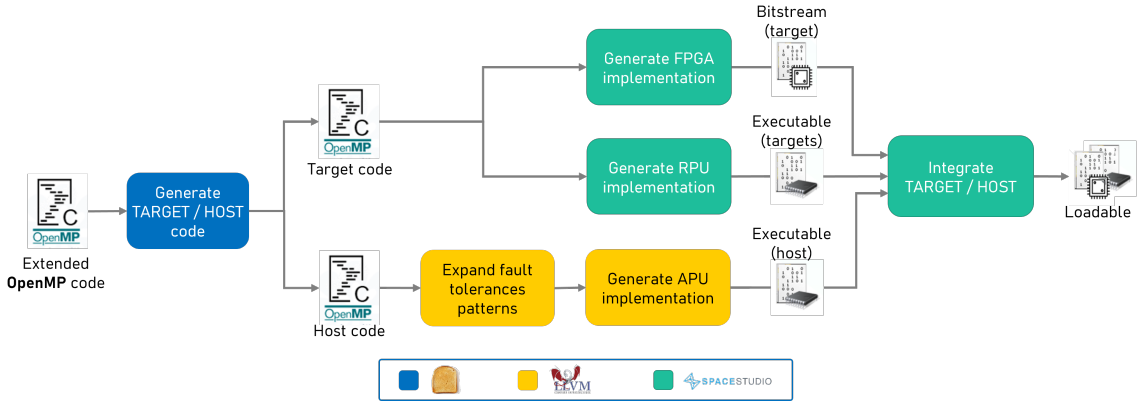


Figure 3: LIONESS software building blocks and general procedure.

3 The LIONESS software framework

The LIONESS software framework builds upon four pre-existing tools, described as follows:

- *Xilinx Vivado 2018.3* (AMD) is the design software for AMD adaptive SoC platforms that enables synthesis and analysis of hardware description designs. *Vivado HLS* assists with transforming C/C++ codes into a register transfer level (RTL) model deployed on Xilinx platforms.
- *SpaceStudio 4.4* (Space Codesign) is a design environment that includes system-level simulation, optimization, and automated FPGA prototyping.
- *TOAST* (IRT Saint Exupéry, Space Codesign) converts OpenMP annotated code into C/C++ code using the SpaceStudio communication application program interface (API).
- *LLVM* (Barcelona Supercomputing Center) is an extended version of the LLVM compilation framework, including fault-tolerance mechanisms through extensions to the OpenMP model and its implementation.

Figure 3 depicts the LIONESS workflow, including inputs and outputs of each tool (Vivado does not appear because it is called from within SpaceStudio). TOAST processes OpenMP annotated applications, dividing the code into a) a host part with each OpenMP `target` directive replaced with the code that sets up the FPGA device using the SpaceStudio API and b) an FPGA accelerator part with the data management from the FPGA side and the computation within the `target` directive. The extended version of LLVM processes the host parts, transforming the code to enable task replication based on user annotations and links the code with the extended OpenMP runtime library supporting replication. Figure 4 shows a possible execution flow of a replicated functionality, where 1) data copies are created for each replica, 2) a functionality is spatially replicated, exposing parallelism, 3) all replicas are synchronized, and 4) a consensus and voting method decides whether a fault occurred and a consolidation function is applied when needed. Finally, SpaceStudio processes the part for the accelerator leveraging Vivado tools to generate bitstreams for the PL, compiles critical parts for the RPU, and links all binaries together, generating an executable that can be loaded into the FPGA platform.

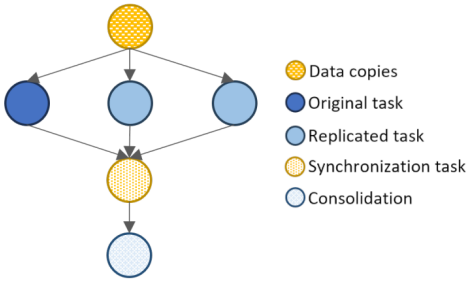


Figure 4: Execution flow of a replicated functionality.

4 Fault tolerance strategy

The Ultrascale+ features numerous mechanisms to ensure safety, including 1) a dual Cortex-R5F core in lock-step, each with its own set of local TCM; 2) a Platform Management Unit (PMU) with a triple modular redundancy (TMR) configuration to manage the Microblazes and error correction (ECC) on the RAM interface; and 3) in the PL, the soft error mitigation (SEM) IP can detect and correct configuration memory errors, enabling partial reconfiguration in case of error.

LIONESS solution for task-level replication complements these mechanisms. However, it presents some constraints. Since a redundant function does not form a strict fault containment region (FCR), there is no guarantee that errors arising will be confined to the redundant component. Fortunately, the time spent in the application is much larger than in the lower layers (e.g., OpenMP runtime). Therefore, the cross-section of the application code with single-event upsets (SEU) is higher than that of the lower layers, so it makes sense to improve the error detection and mitigation for this part.

5 Evaluation plan

LIONESS has designed four different sets of tests that will enable the evaluation of the LIONESS solution on the *resiliency* and the *performance* of the system:

1. *Software benchmarking.* These tests aim to generate a baseline to be able to analyze the benefits of the extensions proposed within LIONESS. The tests will consist

of executing software versions of the four kernels executed, both sequential and parallelized with OpenMP worksharing directives (e.g., `parallel for`).

2. *Performance improvement tests.* These tests aim to evaluate the FPGA offloading to the PL to improve execution time. The tests will consist of launching one or more of the corner detection kernels in the FPGA PL.
3. *Safety improvement tests.* These tests aim to evaluate the task-level replication mechanism and the consolidation to increase the algorithm's safety. These tests will leverage the RPU and the APU.
4. *Performance and safety improvement tests.* These tests aim to evaluate the mixed impact of LIONESS methods on performance and safety holistically. The tests will exploit replication while leveraging the FPGA PL to parallelize replicated tasks.

Additionally, the *programmability* will be evaluated based on the amount and complexity of the modifications needed from the user to exploit the LIONESS software framework.

6 LIONESS: Status and Prospects

LIONESS is a project led by Barcelona Supercomputing Center, participated by Airbus Defence and Space, IRT Saint Exupéry and Space Codesign, and funded by the European Space Agency (ESA). As depicted in the Gantt chart in Figure 5, LIONESS lasts 14 months. It started in February 2024 and will end in March 2025. The project is divided into three work packages (WP) and has three milestones (MS), the first of which has already been successfully reached.

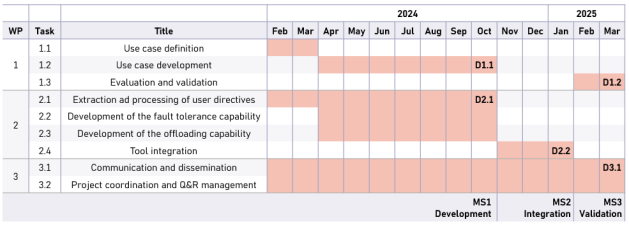


Figure 5: Gantt chart of the LIONESS project.

At the time of writing this paper, each of the methods and tools proposed by LIONESS have been implemented and tested in isolation: a) task-level replication has been implemented in LLVM extending OpenMP directives, b) TOAST and SpaceStudio have been updated to LLVM 17 to support the latest OpenMP directives for offloading tasks, c) the use case has been developed and augmented with the required OpenMP annotations, and d) the platforms have been set up will all the components in their latest version.

During the subsequent phases, the project will perform the integration of the components and will evaluate the use case presented in Section 2.1 using the Xilinx Ultrascale+ platform described in Section 2.2 and following the evaluation plan presented in Section 5. Those interested can follow the progress in the LIONESS LinkedIn profile[1].

[1]Follow LIONESS: https://www.linkedin.com/company/101647278

Acknowledgements

The LIONESS project is carried out under a program of and funded by the European Space Agency under Contract No. 4000141286/23/NL/GLC/ov. The LIONESS team wants to thank the ESA officers who evaluated the idea and the project reviews, who provided invaluable input.

References

[1] M. A. Serrano, S. Royuela, and E. Quiñones, "Towards an OpenMP specification for critical real-time systems," in *14th International Workshop on OpenMP*, pp. 143–159, Springer, 2018.

[2] A. Munera, S. Royuela, R. Ferrer, R. Peñacoba, and E. Quiñones, "Static analysis to enhance programmability and performance in OmpSs-2," in *High Performance Computing: ISC High Performance International Workshops*, pp. 19–33, Springer, 2020.

[3] S. Royuela, A. Duran, M. A. Serrano, E. Quiñones, and X. Martorell, "A functional safety OpenMP for critical real-time embedded systems," in *International Workshop on OpenMP*, pp. 231–245, Springer, 2017.

[4] A. Munera, S. Royuela, and E. Quiñones, "Towards a qualifiable OpenMP framework for embedded systems," in *Design, Automation & Test in Europe Conference & Exhibition*, pp. 903–908, IEEE, 2020.

[5] A. Melani, M. A. Serrano, M. Bertogna, I. Cerutti, E. Quinones, and G. Buttazzo, "A static scheduling approach to enable safety-critical OpenMP applications," in *22nd Asia and South Pacific Design Automation Conference*, pp. 659–665, IEEE, 2017.

[6] M. Samadi, S. Royuela, L. M. Pinho, T. Carvalho, and E. Quiñones, "Time-predictable task-to-thread mapping in multi-core processors," *Journal of Systems Architecture*, vol. 148, p. 103068, 2024.

[7] L. Sommer, J. Korinth, and A. Koch, "OpenMP device offloading to FPGA accelerators," in *28th International Conference on Application-specific Systems, Architectures and Processors*, pp. 201–205, IEEE, 2017.

[8] J. Lopez Trescastro, E. Vassev, D. Hellstrom, and D. Cederman, "An LLVM Backend for LEON Processors," in *DAta Systems in Aerospace*, vol. 732, 2015.

[9] R. Leconte, E. Jenn, G. Bois, and H. Guérard, "Make life easier for embedded software engineers facing complex hardware architectures," in *10th European Congress on Embedded Real Time Software and Systems*, 2020.

[10] G. Nicolescu, G. Bois, M. Benyoussef, J. Boland, and J. Hugues, "Design space exploration: Bridging the gap between high–level models and virtual execution platforms,"

[11] AMD, "Zynq UltraScale+ Device Technical Reference Manual." https://docs.amd.com/r/en-US/ug1085-zynq-ultrascale-trm/Introduction-to-the-UltraScale-Architecture.

Design Space Exploration using Evolutionary Optimisation in Cyber-Physical Systems: A Smart Port Case study

Ken Sharman, Javier Coronel
Instituto Tecnológico de Informática, Valencia, Spain; email: {ken, jcoronel}@iti.es

Sergio Sáez
Instituto Universitario Mixto Tecnológico de Informática,
Universidad Politécnica de Valencia, Spain; email: ssaezba@upv.es

Abstract

This work discusses using evolutionary optimisation algorithms at the design stage of cyber-physical systems. We present a parametrized model for a complex, heterogeneous, distributed cyber-physical system. This model is then used to calculate certain performance metrics of the system which will assist the design engineer. The optimisation of the system structure and its parameters is then performed by an evolutionary algorithm.

Keywords: Cyber-Physical Systems, Evolutionary algorithms, Design Space Exploration.

1 Introduction

A cyber-physical system (CPS) is a computing system that integrates physical processes with computational algorithms and networked communication. Typically a CPS consists of a number of physical devices, the *IoT or device layer*, which both produce and consume data streams in real time to and from processors at the edge of the network. These edge processors are usually low or mid power devices with limited computing capabilities, also called the *edge layer*. Therefore, to implement more computationally demanding algorithms, for example speech recognition, or deep learning, more powerful and/or special purpose nodes can be added as a further logical layer to the network. This is sometimes referred as the *fog layer*. However, as the edge and fog layers are not so clearly distinguished, they are usually referred as the edge layer. Additionally, many applications require access to massive databases or extreme computing power (perhaps for dynamic virtualisation of local nodes). These facilities are normally provided by large data centres, which may be distantly located from the target CPS and accessed over the public network. This is the so called *cloud layer*.

It is also observed that the typical data processing flow in a CPS involves multiple streams which pass through a pipeline of processing stages each of which have differing computational, storage, and sampling rate requirements. Additionally, when computational demands are intense, a CPS processing algorithm might need to take advantage of special purpose processors such as GPUs or use multiple processing cores in parallel.

A final complexity in CPSs is that in some applications the device layer is dynamic. In other words, the number of connected devices and the amount of data they produce or consume varies over time and space. Thus, there is a need for the CPS to take this into account and make sufficient resources available. For economic reasons it would be useful if the CPS itself was dynamic in the sense that could change the number of processing nodes, or the locations of software modules at different times according to actual or predicted demand. This can be achieved using virtual machines in cloud nodes.

Thus, effectively, from a modelling and analysis point of view, a CPS is a complex dynamic system consisting of a number of different types of processing devices, communicating over various links, and running multiple software tasks which will be sending messages amongst themselves. The design and verification of such systems is therefore extremely demanding.

This paper addresses some high level design challenges in these types of CPS. In particular we are concerned here mainly with the *non-functional* properties of the system.

2 Case Study

This work has been carried out in the domain of maritime port infrastructure management. This Smart Port Platform consist of a large set of IoT devices sending information about the position and load of the cranes (STS, RTG, etc.) and port trucks that operate in the container yard. These IoT devices are only active when the corresponding crane or truck is participating in a given port operation. Since these devices are also moving across the container yard, this creates a dynamic infrastructure and the associated communication traffic. All these devices conform the IoT layer.

The edge layer is composed by a set of gateways, distributed throughout the container yard, to which IoT devices connect, together with an optional set of local processing nodes. The processing infrastructure is completed with the main set of

processing nodes that can be allocated in a cloud service provider.

On top of this processing infrastructure a container terminal management software, referred as Terminal Operating System (TOS), has to be deployed. The TOS coordinates, among many other tasks, where the containers has to be placed, the container ships loading and unloading schedule, as well as keeping track of every container location at any moment. This allows the system to provide real time information about the usage of the physical resources (cranes and trucks), the container yard hot-spots, etc. To determine how the TOS software components can be deployed in the smart port computing infrastructure is one of the main objectives of this work.

3 System Modelling

In this section it is shown how to construct an abstract, high-level model of a CPS and then to use that model to calculate certain performance metrics.

3.1 Architecture

We describe a system, S, as a tuple of 3 objects: The *platform model*, P, the *application model*, A, and the *deployment model*, D.

$$S =< P, A, D > \tag{1}$$

The platform model is a representation of the hardware part of the CPS. It is described by a graph structure whose nodes, N, are the processors in the CPS and whose edges, C, are the communications links,

$$P =< N, C > \tag{2}$$

Each processing node, n, in the set N is a list of parameters relevant to that node. These would include, the type of processor, its processing power, memory available, power consumption, scheduling algorithm, and so forth. The communications links, c, in C also have a list of parameters including, obviously, references to the source and destination nodes of the link as well as the type of link, its bandwidth, latency, and so on. In many cases, however, the physical communications links are complex with their own access and scheduling protocols, and shared logical data flows. In such cases we might decompose the link into multiple data paths and a virtual processor to accurately model the scheduling of data packets through the link. Figure 1 shows a diagram of the platform model graph of a typical (but small) CPS.

The software running on the CPS is represented by the application model, A. This is also a graph structure whose nodes, T, are individual software tasks or micro-services. The edges of this graph, M, are the messages that are sent between pairs of software tasks.

$$A =< T, M > \tag{3}$$

The model used in this work considers an event driven system where the execution of a software task is triggered by a message arriving at a (virtual) input port. Also, during the execution of a task it may send messages to other tasks to start their execution too.

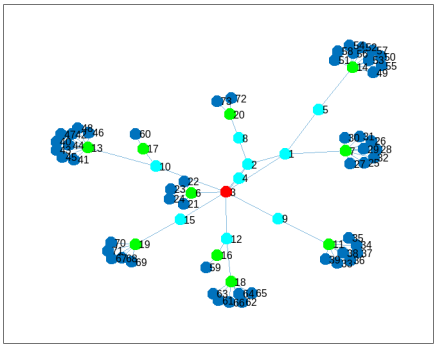


Figure 1: Example of a Platform Model. Blue: IoT devices, Green: gateway nodes, Cyan: local nodes, Red: cloud server

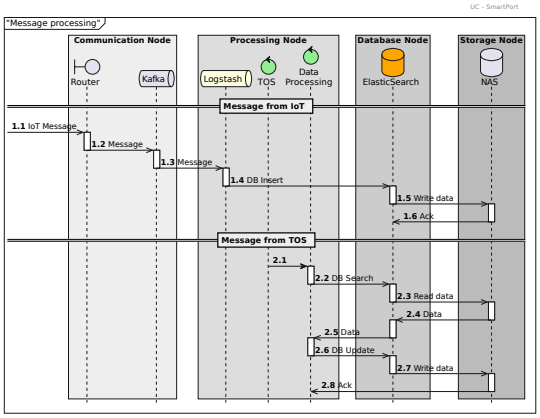


Figure 2: Example of an Application Model

Each task, t, in the application model is assigned a set of parameters including, the worst case execution time of the task when running on a particular type of processing node, the resource requirements of the task (e.g. memory), the deadline of the task (maximum allowable response time), etc. The parameters attached to each message object, m, in the edge set, M, would include source and destination references, the size of the message and other relevant information such as its arrival distribution function (see next section).

Finally, the deployment model, D, describes how the tasks in the application model are mapped to the hardware described in the platform model. This is simply a vector of references to processing nodes: i.e. the n'th element of D indicates that software task n is running on the processing node d_n. Figure 2 shows an example of a sequence diagram with some of the components of the application model.

It is relevant to point out that the system model described above can easily be represented in standard data formats such as SysML [1] or SSP (System Structure and Parametrization [2]. Also, tools such as graphical editors already exist for assisting in the creation and manipulation of these system models (e.g. Modelica [3]).

3.2 System Performance Metrics

It is now considered how to calculate some useful performance metrics of a CPS using the above system model. However, it is important to point out that the assignment of specific performance metrics to a CPS is very domain specific as there are many different ways in which it can be characterised the performance of a CPS. For example, it could interesting to minimise power consumption or financial cost (cloud services charge for number of CPU seconds used), or the goal can be to maximise resource usage in fog processors, to mention but a few.

But perhaps one of the most important metrics is the timing performance of a CPS and in particular how long it takes to complete a chain of software tasks. In many applications it is extremely important that certain tasks are completed within a defined time (a hard deadline). Thus it is needed to determine the expected response times of each software task under different conditions. One way to do this is by simulation and indeed many tools have been developed for this purpose. However, simulation is very computationally intensive and simulations will need to be run for many different environmental conditions further adding to the computational requirements.

Alternatively, analytical methods can be used to calculate bounds for the expected response times of software services. Formal timing analysis methods have been in use for decades, starting with the initial work by Liu and Layland in the 70s [4] and they have been extensively research since then. One popular method which is commonly used today is called *Compositional Performance Analysis* (CPA) [5].

The goal of CPA is to determine upper bounds for the response times of software tasks in a distributed CPS. Recall that in a typical processing node there will be many tasks executing in a time sharing environment and that these tasks will be controlled by a scheduling algorithm (e.g. priority driven). Thus there is interference between different tasks running on a processor: the execution of a lower priority task can potentially be interrupted by a higher priority task and thus the *response time* of the lower priority task will be increased. For a single processing node, given knowledge of the processor's scheduling algorithm, it is reasonably straightforward to calculate, using an iterative algorithm, upper bounds for the response time of each task [6]. This is called *local response time analysis* (l-RTA).

To do this, however, it is needed to define an operating environment for the CPS in terms of the temporal aspects of when and how each software task is triggered (or released) for execution. Usually, in a CPS, a task is released either on a time-triggered basis (e.g. periodic activation), or on an event-triggered basis (e.g. due to the completion of another task or by an event emitted by and external device). In both cases we can define an event model for each task in terms of a pair of so-called *distance functions* $\delta_i^+(n)$ and $\delta_i^-(n)$. The maximum distance function, $\delta_i^+(n)$ describes the maximum time distance between any n consecutive activation (or termination) events of software task t_i, while the minimum distance function $\delta_i^-(n)$ describes the minimum time interval between any n consecutive activation (or termination events) of task t_i.

The CPA algorithm proceeds by initially using these distance functions at the edge of the network along with an l-RTA of the first software tasks triggered by the IoT connected devices. This initial local analysis allows the calculation of the distance functions of the events which will trigger subsequent software tasks defined by the connections, M, in the application model A. In other words the output distance functions of a predecessor task now become the input distance functions for the successor tasks. This process is repeated until the response times for all tasks have been computed. However, due to potential feedback loops this process must be iterated several times until convergence is achieved (or not). If convergence is not achieved it is concluded that the system is unstable and it does not have sufficient computing resources to execute all the software tasks.

At the end of the CPA procedure upper and lower bounds are obtained for the response times of each task. This is a very important performance metric as it not only allows us to determine if the system is stable or not, but it also enables the calculation of other useful metrics such as maximum data throughput or response latencies.

4 Evolutionary Optimisation

The development of a simple system model along with the calculation of performance metrics now allows the possibility of searching for a suitable system configuration to satisfy the defined requirements, or to optimise a certain performance metric. In many applications though there may be two or more competing requirements (e.g. speed of response versus financial cost). A further complication is that the performance metrics which we described above are complex calculations. They are non-differentiable functions, which therefore precludes many traditional optimisation methods. Therefore it is proposed to use so called evolutionary algorithms for this optimisation task. These are a class of search algorithms which have proven themselves to be extremely powerful for solving multi-objective optimisation problems under multiple non-linear constraints. A good introduction to evolutionary optimisation is given in the book by Eiben and Smith [7]. In fact, there is some history of using evolutionary algorithms for optimising CPS and a good survey of these is presented in the paper by Guerrero et al [8].

The great advantage of evolutionary algorithms is that they are a class of *black-box* optimisation methods. That is they are useful when the objective function is highly complex and/or expensive to calculate. In addition they are well suited to a range of widely different problem types, including those with continuous, discrete, or combinatorial domain variables. Evolutionary algorithms are inherently parallelizable, as multiple candidate solutions can be evaluated concurrently. This allows for efficient exploration of the search space, speeding up the optimization process, especially in distributed or parallel computing environments.

Additionally, evolutionary algorithms tend to be robust in noisy or uncertain environments. Because they maintain a

diverse population of solutions and use probabilistic methods for selection, crossover, and mutation, they are less likely to get stuck in local optima and more capable of adapting to changes in the environment or problem constraints.

There are four key tasks required in building a successful evolutionary optimiser. The first of these is to define a suitable *fitness function* that is to be optimised (maximised or minimised). In the application considered in this paper this fitness function likely to be a combination of two (or more) of some of the performance metrics which we described in section 2 above.

The second key task is to define the parameters to be adjusted during the optimisation process. This set of parameters is sometimes called the *gene*. In the system model presented above there are a wide range of possible parameters that may be chosen for optimisation according to the underlying problem at hand. These include, real-valued, continuous variables such as, for example, the speed of a particular processing node or the bandwidth of a communications link, discrete variables such as the number of cores at a processing node, or topological variables such as the deployment of software tasks to particular processing nodes (i.e. the D object in system model definition given in equation 1 above).

It is also possible to parametrize the hardware architecture within the gene. For example, the part of the system gene might contain an index into a table of different platform objects, P, in the system model defined in of equation 1. In this way the optimiser can evaluate different system configurations during the evolution.

The third key task is to define constraints for the variables that compose the elements of an individual candidate's gene. For example, it could be that a certain software task is only allowed to run on certain processing nodes. This constraint would therefore be a restriction of the values of the D object in the system model. Other constraints might be that the resources of a processing node are not exhausted or that the bandwidth of a communications link is not exceeded.

The final task is constructing an evolutionary optimiser is to choose the parameters of the optimisation algorithm itself. This is not an easy task as a typical algorithm may have dozens of parameters that need to be chosen and which will affect performance. Indeed, the problem of selecting suitable optimiser parameters in an area of active research [9].

Note that, when there is more than one performance objective the result of the optimisation process is not a single optimum system. Rather, the optimiser will produce a set of possible system solutions. In multi-objective optimization, the Pareto front is the set of all Pareto efficient solutions. The concept is widely used in engineering. It allows the designer to restrict attention to the set of efficient choices, and to make trade-offs within this set, rather than considering the full range of every parameter.

As a brief example of evolutionary optimisation in CPS an output of an evolutionary optimiser is shown in figure 3. In this example, there were two objectives being optimised: the average latency between software tasks, and the financial cost

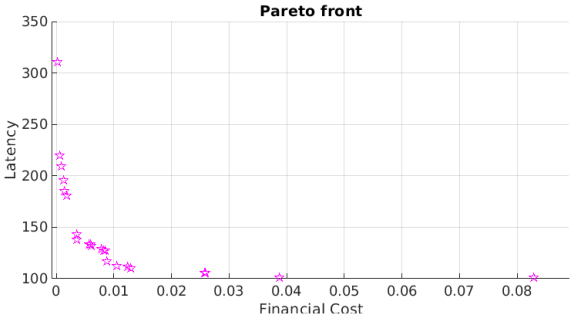


Figure 3: Pareto Front After Optimisation for a Two Variable Problem

of the overall system. As can be seen from the Pareto front, there is a clear trade-off between the expense of the system and it's overall speed.

5 Conclusions

This paper presented an introductory overview of using a multi-objective evolutionary algorithm to assist an engineer in the high level design of a complex cyber-physical system. It was shown how an abstract model of the system can be parametrized and used to calculate some useful non-functional performance metrics which can then be optimised using a multi-objective evolutionary algorithm.

There are, however, significant challenges and opportunities for further research in this area. One problem is the computational cost of the procedure. A typical evolutionary optimiser might involve an working with a population of thousands of individual systems evolving over perhaps hundreds or thousands of generations. This is a significant computational requirement. However, it is pointed out that evolutionary algorithms are extremely well suited to parallel implementations so this would be one way of reducing the computational time.

Another challenge would be to investigate the possibilities of implementing the optimisation process at run-time rather than at design time. In this way the system could dynamically adjust its configuration to account for differing environmental conditions at different times.

Acknowledgements

Special thanks to all AIDOaRt (Artificial Intelligence for DevOps at Run-Time) consortium members who are currently working on this project. The AIDOaRt project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, and Spain.

References

[1] SysML.org, "Sysml open source project: What is sysml." `http://sysml.org`.

[2] M. Association", "Ssp - system structure & parameterization." `http://ssp-standard.org`.

[3] M. Association", "Modelica - model complex systems more effectively." `http://modelica.org`.

[4] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, p. 46–61, jan 1973.

[5] R. Hofmann, L. Ahrendts, and R. Ernst, "CPA: Compositional Performance Analysis," in *Handbook of Hardware/-Software Codesign* (S. Ha and J. Teich, eds.), pp. 721–751, Dordrecht: Springer Netherlands, 2017.

[6] J. Palencia and M. Harbour, "Offset-based response time analysis of distributed systems scheduled under EDF," in *15th Euromicro Conference on Real-Time Systems, 2003. Proceedings.*, (Porto, Portugal), pp. 3–12, IEEE Comput. Soc, 2003.

[7] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer Publishing Company, Incorporated, 2nd ed., 2015.

[8] C. Guerrero, I. Lera, and C. Juiz, "Genetic-based optimization in fog computing: Current trends and research opportunities," *Swarm and Evolutionary Computation*, vol. 72, p. 101094, 05 2022.

[9] A. Eiben and S. Smit, "Evolutionary algorithm parameters and methods to tune them," *Autonomous Search*, pp. 15–36, 01 2012.

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. Johan Nielsen
Email: jon@jonsoft.dk

Ada-Deutschland

Dr. Hubert B. Keller CEO
ci-tec GmbH
Beuthener Str. 16
76139 Karlsruhe
Germany
+491712075269
Email: h.keller@ci-tec.de
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

Ada-Europe Sponsors

Ada Edge

27 Rue Rasson
B-1030 Brussels
Belgium
Contact: Ludovic Brenta
ludovic@ludovic-brenta.org

AdaCore

46 Rue d'Amsterdam
F-75009 Paris
France
sales@adacore.com
www.adacore.com

AdaLabs
innovate.all

506 Royal Road
La Caverne, Vacoas 73310
Republic of Mauritius
Contact: David Sauvage
david.sauvage@adalabs.com
www.adalabs.com

ADA LOG

2 rue Mozart
92110 Clichy
France
Contact: Jean-Pierre Rosen
rosen@adalog.fr
www.adalog.fr/en/

Deep Blue Capital

Jacob Bontiusplaats 9
1018 LL Amsterdam
The Netherlands
Contact: Wido te Brake
wido.tebrake@deepbluecap.com
www.deepbluecap.com

Ellidiss Technologies

24 Quai de la Douane
29200 Brest, Brittany
France
Contact: Pierre Dissaux
pierre.dissaux@ellidiss.com
www.ellidiss.com

EUROCITY

Rue Marie de Bourgogne 52
1000 Brussels
Belgium
Contact: Emma Claus
Emma.Claus@eurocity.be
www.eurocity.com

KONAD
Software for Control and Administration

In der Reiss 5
D-79232 March-Buchheim
Germany
Contact: Frank Piron
info@konad.de
www.konad.de

PTC® Developer Tools

3271 Valley Centre Drive,Suite 300
San Diego, CA 92069
USA
Contact: Shawn Fanning
sfanning@ptc.com
www.ptc.com/developer-tool

SYSADA

Enterprise House
Baloo Avenue, Bangor
North Down BT19 7QT
Northern Ireland, UK
enquiries@sysada.co.uk
sysada.co.uk

systerel
Safe real-time solutions

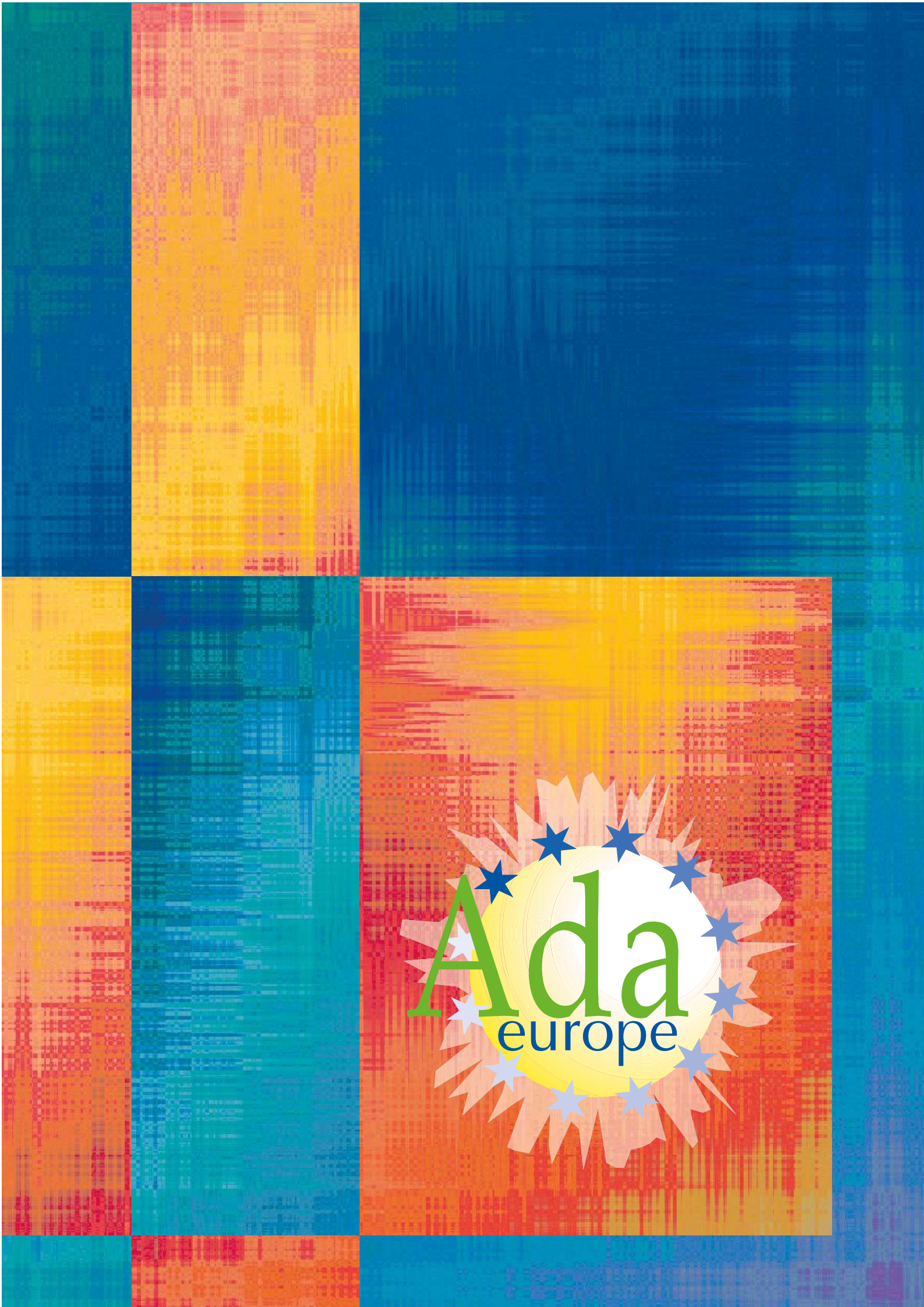
1115 Rue René Descartes
13100 Aix en Provence
France
Contact: Patricia Langle
patricia.langle@systerel.fr
www.systerel.fr/en/

Tid rum

Tiirasaarentie 32
FI 00200 Helsinki
Finland
Contact: Niklas Holsti
niklas.holsti@tidorum.fi
www.tidorum.fi

WhiteElephant

Bachstrasse 51
8200 Schaffhausen
Switzerland
Contact: Ahlan Marriott
admin@white-elephant.ch
www.white-elephant.ch


Ada
europe