The journal for the international Ada community

Ada User Journal

Volume 46
Number 2
June 2025

Editorial 75
Quarterly News Digest 76
Conference Calendar 99
Forthcoming Events 106

Articles from the AEiC 2025 Work-in-Progress Session

N. Costa, J. Cecílio, R. Salgueiro, M. Rosa, D. Domingos
LLM-based Framework for Email Classification and Phishing Detection 108

G. Calliet
Archaeology for the Future - Ada on OpenVMS 112

T. Carvalho, L. M. Pinho, P. Paiva
Development and Management of Digital Twins for Energy Monitoring Systems 116

A. Geraldo, J. Cecílio, P. M. Ferreira, A. Casimiro, A. Oliveira de Sá
A Comparative Study of Reliable Predictive Models for Energy-Efficient MPC in Buildings 120

J. Kohl, G. Muck, G. Jäger, S. Zug
Dynamic System Model Generation for Online Fault Detection and Diagnosis of Robotics Systems 124

J. K. Fichte, T. Philipp
Verified SAT Redundancy Checking in SPARK 129

Produced by Ada-Europe

Editor in Chief

António Casimiro — University of Lisbon, Portugal
AUJ_Editor@Ada-Europe.org

Ada User Journal Editorial Board

Luís Miguel Pinho, *Associate Editor* — Polytechnic Institute of Porto, Portugal, *lmp@isep.ipp.pt*

Jorge Real, *Deputy Editor* — Universitat Politècnica de València, Spain, *jorge@disca.upv.es*

Patricia López Martínez, *Assistant Editor* — Universidad de Cantabria, Spain, *lopezpa@unican.es*

Dirk Craeynest, *Events Editor* — KU Leuven, Belgium, *Dirk.Craeynest@cs.kuleuven.be*

Alejandro R. Mosteo, *News Editor* — Centro Universitario de la Defensa, Zaragoza, Spain, *amosteo@unizar.es*

Ada-Europe Board

Luís Miguel Pinho (President) Polytechnic Institute of Porto	Portugal
Dirk Craeynest (Vice-President) Ada-Belgium & KU Leuven	Belgium
Dene Brown (General Secretary) SysAda Limited	United Kingdom
Ahlan Marriott (Treasurer) White Elephant GmbH	Switzerland
António Casimiro (Ada User Journal) University of Lisbon	Portugal

Ada-Europe General Secretary

Dene Brown
SysAda Limited
Signal Business Center
2 Innotec Drive
BT19 7PD Bangor
Northern Ireland, UK

Tel: +44 2891 520 560
Email: Secretary@Ada-Europe.org
URL: www.ada-europe.org

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 46
Number 2
June 2025

Contents

	Page
Editorial Policy for Ada User Journal	74
Editorial	75
Quarterly News Digest	76
Conference Calendar	99
Forthcoming Events	106
Articles from the AEiC 2025 Work-in-Progress Session	
N. Costa, J. Cecílio, R. Salgueiro, M. Rosa, D. Domingos "*LLM-based Framework for Email Classification and Phishing Detection*"	108
G. Calliet "*Archaeology for the Future - Ada on OpenVMS*"	112
T. Carvalho, L. M. Pinho, P. Paiva "*Development and Management of Digital Twins for Energy Monitoring Systems*"	116
A. Geraldo, J. Cecílio, P. M. Ferreira, A. Casimiro, A. Oliveira de Sá "*A Comparative Study of Reliable Predictive Models for Energy-Efficient MPC in Buildings*"	120
J. Kohl, G. Muck, G. Jäger, S. Zug "*Dynamic System Model Generation for Online Fault Detection and Diagnosis of Robotics Systems*"	124
J. K. Fichte, T. Philipp "*Verified SAT Redundancy Checking in SPARK*"	129
Ada-Europe Associate Members (Ada Organizations)	136
Ada-Europe Sponsors	Inside Back Cover

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

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

Original Papers

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

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

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

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

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

Proceedings and Special Issues

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

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

News and Product Announcements

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

Reprinted Articles

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

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

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

Editorial

The International Conference on Reliable Software Technologies has been running since 1996, and this year its 29th edition took place in Paris, France. The Covid-19 pandemic forced us to cancel the 2020 conference and switch to an online-only edition in 2021. Almost in parallel, starting in 2019, the conference adopted a journal-first modus operandi, introducing a Journal Track for papers accepted, or likely to be accepted, for publication in a Special Issue of the Journal of Systems Architecture (Elsevier). Despite the pandemic years, these special issues continued to appear annually, and the seventh is now close to completion, featuring once again around six high-quality research papers. Looking back, I believe the decision made in 2019 was both wise and crucial for the long-term sustainability of the conference.

While Journal Track papers presented at AEiC are published in the special issue, the Ada User Journal gives us the opportunity to feature other valuable contributions from the conference. In this issue, we begin publishing articles from the AEiC 2025 Work-in-Progress Session. The six included papers span a wide range of topics related to software and systems reliability. Readers will find work on email security, Ada history, modelling and digital twins for energy monitoring and management, reliability of robotics systems, and verification techniques in Ada/SPARK. The contributions come from authors based in Portugal, Germany, Sweden and France.

As usual, this issue also includes the News Digest section, prepared by Alejandro R. Mosteo, as well as the Conference Calendar and Forthcoming Events sections, prepared by Dirk Craeynest. You will also find, on pages 106–107, a preliminary Call for Contributions to the 30th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2026), which will be held next year in Västerås, Sweden. Next year's conference will again feature a rich set of contributions, including the usual journal, industrial and work-in-progress tracks, but also a regular (proceedings) track. As the journal track model of the conference consolidates, the new track will allow a broader scope for scientific contributions with proceedings being published in the OASIcs Open Access Series in Informatics.

Antonio Casimiro
Lisboa
June 2025
Email: AUJ_Editor@Ada-Europe.org

Quarterly News Digest

Alejandro R. Mosteo

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

Contents

Preface by the News Editor	76
Obituaries	76
Ada-related Events	77
Ada-related Resources	81
Ada-related Tools	83
Ada-related Products	89
References to Publications	89
Ada and Other Languages	89
Ada Practice	90
Ada Frontiers	95
SPARK/Ada	97

[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,

I am saddened to have to report the passing of esteemed Ada personality Simon J. Wright. Widely regarded as the go-to person for anything Ada-on-macOS, he will be sorely missed. Expressions of appreciation can be found in threads both in comp.lang.ada [1] and in forum.ada-lang.io [2].

Not to dwell solely on sad news, there is a fascinating thread on Ada 83 compilers and the possibility of reviving some of them [3]. Also, the next Ada iteration is in the works and preliminary ideas have been gathered, which are discussed here [4].

Sincerely,
Alejandro R. Mosteo

[1] "Simon J. Wright", in Obituaries

[2] "Simon J. Wright - RIP", in Obituaries

[3] "Open Source Ada 83 Compilers", in Ada-related Tools

[4] "Plan for Next Version of International Ada Standard", in Ada Frontiers

Obituaries

Simon J. Wright - RIP

From: Jesper-Quorning
Subject: Simon J Wright - Rip
Date: Thu, 29 May 2025 17:16:21 +0000
Newsgroups: forum.ada-lang.io

Saddened to read this:
Memorial to Simon J Wright [1]

[1] https://github.com/R34P3R-H09/RIP-SIMON-WRIGHT

From: OneWingedShark
Date: Thu, 29 May 2025 17:47:08 +0000

Yes, this is sad news.

From: Irvise
Date: Thu, 29 May 2025 21:03:51 +0000

He was a joyous, prolific Ada programmer, he was the soul of GNAT in Mac/Apple devices… I am glad that we could have some chats over at the monthly meetup… This is very sad. Posthumous thank you for your outstanding work.

RIP, you will be missed.

From: AJ-Ianozi
Date: Thu, 29 May 2025 21:17:57 +0000

This is heartbreaking news. Simon has been a vital and instrumental member of the Ada community for years, and someone who I respect and look up to highly. I would never have been able to use GNAT on my Mac without him, and all of my interactions with him have been so positive over the years. Rest in Peace.

From: khv2online
Date: Thu, 29 May 2025 21:28:44 +0000

Rest in Peace Mr. Wright

From: Lucretia
Date: Thu, 29 May 2025 21:35:57 +0000

R.i.P. fellow Englander.

From: gast
Date: Thu, 29 May 2025 21:38:10 +0000

Rest in Peace, Mr. Wright.

From: kevlar700
Date: Thu, 29 May 2025 21:45:06 +0000

Certainly one of the best. Rest in peace.

From: jere
Date: Thu, 29 May 2025 22:08:26 +0000

He was one of my favorite folks in the community. So nice, helpful, cared about others. I'll miss him.

From: streaksu
Date: Thu, 29 May 2025 23:00:01 +0000

All interactions I've had with Simon were amazing, and he always was willing to put his time and soul into helping people new and old in the community. Condolences to his family and close ones. Thanks for everything.

From: Micronian2
Date: Thu, 29 May 2025 23:37:27 +0000

I am so shocked and saddened by the news of Simon's passing. He was an invaluable member of our community, offering guidance and support for many years. His loss leaves a significant void. If he has family, I wish them well during this difficult time.

From: brownts
Date: Fri, 30 May 2025 01:08:53 +0000

This is very sad news indeed. Simon, you will be missed. He was always so enthusiastic and helpful. A big loss for the community. Rest in peace.

From: Ret_Build_Engineer
Date: Fri, 30 May 2025 02:36:02 +0000

I too appreciate greatly the positive impact Simon had on me. Very patient, pleasant, positive, extremely knowledgeable and helpful.

Is there a widow and/or family whom we could inform of how much he will be missed? That might be comforting to them.

On a technical note, is there someone who can at least partially fill his shoes with regards to Ada compiler support for Apple Silicon?

From: docandrew
Date: Fri, 30 May 2025 04:24:02 +0000

This is really shocking and heartbreaking. Simon was really a gentleman, eager to help and full of wisdom and knowledge. He'll be missed. Rest in peace Simon!

From: heharkon
Date: Fri, 30 May 2025 05:39:19 +0000

Ohh… Really sad news. I've had some small interactions with him over the years, while learning Ada, and he was always the most kind and knowledgeable person and a helping hand.

From: captain-haddock17
Date: Fri, 30 May 2025 07:17:46 +0000

Oh This is heartbreaking news! Simon showed me how to build GNAT on my Mac (and also could do it on FreeBSD).

I owe him a lot for his regular GNAT builds deliveries for macOS!

Thank you Simon for your writings and answers in the Ada discussion forums, with always a relevant and kind response.

He was certainly a good person.

To his family,

Rest in Peace

William

From: DirkCraeynest
Date: Fri, 30 May 2025 09:51:27 +0000

So sorry to read this.

Simon really was very important for the Ada community, not in the least for his work supporting Ada on Apple systems. A very friendly and helpful person too, and active in many Ada-related forums.

I was lucky to have met Simon IRL, though it was a long time ago: at the Ada-Europe conferences in Vienna (2002) and York (2005); we shared a cab to the airport in Vienna after the conference…

Reminds us once more that we're not immortal…

From: mosteo
Date: Fri, 30 May 2025 10:34:13 +0000

He was a great guy and a great programmer, a true guru in the best sense of the word. He always went beyond expectations in all my interactions with him, generally due to some quirk of Alire on macOS.

He showed us that you can be on top of your game until the very end, which I find comforting. Godspeed, Simon.

From: NumberSix
Date: Fri, 30 May 2025 15:22:22 +0000

I had used its simple and straightforward EWS web server as the basis for a telemetry system. Good, readable code, easily tweakable, accompanied by good documentation.

Perhaps this is a good time to say that, just as Mr. Wright was a great and helpful man, the Ada community he had served is also a great and helpful community, led by good people.

That, too, is a great merit.

Rest in Peace, Mr. Wright.

From: zmower
Date: Fri, 30 May 2025 21:20:56 +0000

I used to work with Simon before he retired. Lovely man, always helpful. Carried on in the same vein after he retired. I'll miss him.

Chris

Simon J. Wright

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Simon J. Wright
Date: Sat, 31 May 2025 08:34:23 +0000
Newsgroups: comp.lang.ada

Simon J. Wright passed away recently.

Simon is well known through his work supporting GNAT on the Mac, his participation in many Ada forums, and by being a very active and helpful member of the Ada community until the end.

https://forum.ada-lang.io/t/simon-j-wright-rip

He will be missed...

Dirk

From: Moi <findlaybill@blueyonder.co.uk>
Date: Sat, 31 May 2025 23:08:02 +0000

This is very sad news indeed.

Bill F.

From: Niocláisín Cóilín De Ghlostéir <spamassassin@irrt.de>
Date: Sun, 1 Jun 2025 10:50:00 +0000

Many Adaists began during this week to express sadness over Simon J. Wright's death but following hyperlinks via a first hyperlink on that Discourse Ada-forum webpage from May 29th, 2025 gives the notion that he died before April 3rd, 2025.

I noticed no alert on Team Ada and USENET and that Discourse Ada forum that Ms. Charlene Roberts-Hayden died on May 11th, 2025. Cf. Bryan Marquard, "Charlene Roberts-Hayden, pioneering Black woman in computer programming, dies at 86", "The Boston Globe",

https://www.BostonGlobe.com/2025/05/18/metro/charlene-roberts-hayden-computer-programming-pioneer-passes-away

which David Emery hyperlinks to from the LinkedIn group that is called Ada Programming Language since circa May 31st, 2025.

She wrote to Hal Hart via Team Ada in 1997:

"Hal,
I strongly agree with your comments. I know here in the Boston area, on various committees I participate on, I have heard nothing but distress and regre from the proponents of Ada who know the worth of the only highly reliable HOL in existence today and sarcasm from Ada's opponents, "even the DoD has given up on Ada". I think that without the Ada mandate, given the popular, existing languages unreliability and the software explosion far surpassing that of the 1980's the DoD is headed for an even more catastrophic software revolution than that which preceded and was the impetus for the Ada language."

From: Stéphane Rivière <stef@genesix.org>
Date: Sun, 1 Jun 2025 12:55:59 +0000

Niocláisín, thanks for your comments

Some related links (no paywall)

https://www.blackcatholicmessenger.org/charlene-roberts-hayden-obit/

https://hart-site.net/Purdue-CS-Panel-Reunion/WhateverHappenedTo/HalHart.html

> She wrote to Hal Hart via Team Ada in 1997: [...]

This was very true and has been verified.

Ada-related Events

Ada Monthly Meetup, 5th of April 2025

From: Irvise
Subject: Ada Monthly Meetup, 5th of April 2025
Date: Wed, 2 Apr 2025 21:36:12 +0000
Newsgroups: forum.ada-lang.io

Are there any topics that someone would like to share? I have received no requests.

Also, this meetup will be the last in a long time. In may I will not be available and June's meetup will be cancelled in favour of the AEiC 2025 Ada Developers Workshop, just saying

From: Irvise
Date: Sat, 5 Apr 2025 14:57:38 +0000

We just had the meetup and here are the topics that were discussed:

- @Max presented the following topics:
  - He has been working with https://notebooklm.google/ in order to create Podcast style conversations that cover Ada topics. He has already uploaded one chapter/sample and he would like to have feedback on it! https://www.youtube.com/watch?v=mxlTFRkgRro
  - He has also restarted work on Ada support for the Yocto Project https://www.yoctoproject.org/ A Linux build system geared towards small custom images for embedded and small devices. His work is present in GitHub - reznikmm/meta-ada at topic/hack [1]
  - He also pointed out the webinar from AdaCore, where they present the new improvements to their lineup of products and expected language improvements https://www.youtube.com/watch?v=mu4DEAHZeko
- Additionally to the video, anybody can have discussions regarding improvements to the language as proposed by AdaCore in their RFC GitHub repo GitHub - AdaCore/ada-spark-rfcs: Platform to submit RFCs for the Ada & SPARK languages [2] If you would prefer to see what the ARG is up-to and partake in the discussions, please, refer to Ada Rapporteur Group [3] and the Community Input tab.
- Fernando showcased the changelog for GCC/GNAT 15. It will be made public next week.

- A reminder on the Ada Developer Workshop call for proposals was made. It is still open for the next few days! Remember you can make a proposal for an online talk! https://www.ada-europe.org/conference2025/index.html
  - The cost of the Workshop is still not public pending some discussions for some sponsorships.

Talk to you soon.

[1] https://github.com/reznikmm/meta-ada/tree/topic/hack

[2] https://github.com/AdaCore/ada-spark-rfcs

[3] https://arg.adaic.org/home

Ada Developers Workshop @ AEiC, Speakers, Topics and Price

From: Irvise
Subject: Ada Developers Workshop @ AEiC, Speakers, Topics and Price
Date: Thu, 8 May 2025 20:40:49 +0000
Newsgroups: forum.ada-lang.io

Dear community,
It pleases us to announce the list of speakers and topics for the upcoming Ada Developers Workshop, which will be taking place on June 13th, in Paris, as part of the Ada-Europe international Conference on Reliable Software Technologies [1]. The list below does not represent the schedule, which will be published in the main website when ready:

- @jgrivera67: Writing Embedded Ada Applications on Zephyr
- @stcarrez: Automating License Identification with SPDX-Tool in Ada
- Pascal Pignard UXStrings: a Unicode and dynamic length string library for Ada
- Xavier Petit, Stèphane Rivière: A KISS Ada GUI
- @LionelDraghi: Using natural language for test specification, is that really wise?
- @zertovitch: Writing a competitive BZip2 encoder in Ada from scratch in a few days
- @Fabien.C: Property Based Testing in Ada: the missing 10%
- @Fabien.C: Let's Write a Safety Monitor for a Mars Rover!

Also, we are happy to announce that thanks to the generous support from AdaCore, Ada-France and Ada-Belgium, we have been able to reduce the cost of attendance in person to 20€ for early registration and 35€ for late ones (up-to 20 people). As a reminder, online participation will be possible at no cost, however, we ask you to register, in order to keep some traceability of the number of participants and have clean communications.

The registration page [2] will open soon. There is also already a list of available accommodation places [3] for the conference. Of course, you are all more than welcome to join us however is best to you!

The Ada Developers Workshop team.

[1] https://www.ada-europe.org/conference2025/workshop_adadev.html

[2] https://www.ada-europe.org/conference2025/registration.html

[3] https://www.ada-europe.org/conference2025/accommodation.html

From: DirkCraeynest
Date: Fri, 9 May 2025 06:15:03 +0000

Thanks for distributing this info, Fer.

For the record. You wrote:

> Thanks to the generous support from AdaCore, Ada-France and Ada-Belgium, we have been able to reduce the cost of attendance in person to 20€ for early registration and 35€ for late ones (up-to 20 people).

This should be (diffs emphasized): thanks to the generous support from AdaCore, /Ada-Europe/, Ada-France and Ada-Belgium, we have been able to reduce the cost of attendance in person to /25€/ for early registration and 35€ for late ones (up-to 20 people).

From: DirkCraeynest
Date: Sat, 14 Jun 2025 12:05:47 +0000

Members of the #AdaProgramming language community met yesterday, in a "tropical" Paris and online, at the 2nd Ada Developers Workshop #AdaDevWS held at the 29th #AdaEurope International Conference on Reliable Software Technologies #AEiC2025. "And there was much rejoicing!"

Slides and video recordings of all technical presentations will be uploaded to the #AdaDevWS webpage at

https://www.ada-europe.org/conference2025/workshop_adadev.html

Ada-Europe Int. Conf. Reliable Software Technologies, AEiC 2025

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe Int.Conf. Reliable Software Technologies, AEiC 2025
Date: Mon, 12 May 2025 20:12:34 +0000
Newsgroups: comp.lang.ada

Call for Participation

29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025)

10-13 June 2025, Paris, France

www.ada-europe.org/conference2025

*** Early registration DEADLINE May 27 ***

** Extensive info and registration online ***

*** Add tutorials and/or a workshop to your conference registration ***

#AEiC2025 #AdaEurope #AdaProgramming

Organized by Ada-Europe and Ada-France, in cooperation with ACM SIGPLAN, supported and sponsored by Embedded France, AdaCore, Emenda, Systerel, Scitools, Pacte Novation, Institut Frederik Bull, and Mines Paris

---

Preparations for AEiC 2025, the 29th Ada-Europe International Conference on Software Reliable Technologies, are well underway!

Come to the Ada-Europe conference in Paris, experience a packed program in an exciting town, benefit from tutorials on Tuesday, join a workshop on Friday, enjoy the social events and some sightseeing!

Register now: discounted fees until May 27!

<http://www.ada-europe.org/conference2025/registration.html>

Extra conference sponsorship allows for an extremely low 25 EUR fee for the Ada Developers Workshop on Friday!

See below for an overview, and visit our website for more details about the conference program, registration fees, social events and many more.

General Information

The 29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025) returns after 11 years to Paris, France. The conference schedule comprises keynote and invited talks, a journal track, an industrial track, a work-in-progress track, a vendor exhibition, parallel tutorials, and satellite workshops. There will be time for networking during breaks and lunches, as well as various social events.

AEiC 2025 is the latest in a series of annual international conferences started in the early 80's, under the auspices of Ada-Europe, the international organization that promotes knowledge and use of the Ada programming language and reliable software in general, into academic education and research, and industrial practice.

The Ada-Europe series of conferences has over the years become a leading international forum for providers, practitioners and researchers in reliable software technologies. These events highlight the increased relevance of Ada in general and in safety- and security-critical systems in particular, and provide

a unique opportunity for interaction and collaboration between academics and industrial practitioners.

The 2025 edition of the conference continues a number of important innovations started in previous years:

- reduced conference registration fee for all authors;
- extra low registration fee Ada Developers workshop;
- journal-based open-access publication model for peer-reviewed papers;
- compact program with two core days (Wednesday & Thursday);
- tutorials on Tuesday, followed by welcome event for all participants;
- workshops on Friday, optional chill event on Thursday evening.

Overview of the Week

- Tue 10: 4 half-day tutorials, 1 full-day tutorial, welcome reception;
- Wed 11: core technical program, conference banquet;
- Thu 12: core technical program, post conference chill-out;
- Fri 13: 3 full-day workshops.

Extensive information on AEiC 2025 will be on the conference website, such as an overview of the program, the list of accepted papers and presentations, and descriptions of workshops, tutorials, keynote and invited presentations, and social events. Also check the conference site for registration, accommodation and travel information. The Final Program brochure will be available there as well.

Venue

<http://www.ada-europe.org/conference2025/venue.html>

The conference will take place at Mines Paris, a founding member of Université PSL, a leading French engineering school and the French leader institution in research partnerships. 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. If you can stay over before or after the conference, there's a lot to see. Check the Practical Information section of the conference website for more info.

Invited Speakers

<http://www.ada-europe.org/conference2025/keynotes.html>

This year the conference will once more feature keynotes and invited talks. All will address topics of relevance in the conference scope, with time for questions and answers.

- On Wed morning, June 11, a keynote talk by Stefano Zacchiroli, of Télécom Paris, France, on "Building a Safer Open Source Supply Chain with Software Heritage, the Great Library of Source Code".
- On Wed afternoon, June 11, an invited talk by Tullio Vardanega, from University of Padua, Italy, on "Language Ownership: Key Moments in the Lifetime of Ada".

Conference Core Composition

The core conference program features three distinct types of technical presentations, with different duration, in addition to the keynote and invited talks: journal-track talks (25 minutes), industrial-track talks (20 minutes), vendor presentations (15 minutes), work-in-progress-track talks (10 minutes).

All papers presented in the journal track, the industrial track and the work-in-progress track have undergone peer review. Presentations are combined into by-theme and not by-track sessions, in order that authors and participants alike enjoy all flavors of the program in a mixed as opposed to segregated combination.

The list of accepted papers and presentations, and the detailed schedule, will be announced shortly.

Tutorials

<http://www.ada-europe.org/conference2025/tutorials.html>

A full-day tutorial is offered on Tuesday 10th:

- "Developing Bare-metal Embedded Software in SPARK Ada for 64-bit ARM Platforms", by J. Germán Rivera, Tesla, USA.

In parallel, four half-day tutorials are scheduled:

- "Multiple Double and Multiword Arithmetic", by Jan Verschelde, University of Illinois at Chicago, USA;
- "Unleash the Power of Ada Generics", by Gautier de Montmollin, Ada-Switzerland, Switzerland;
- "Turning your Programming Language into a Modeling Language", by S. Tucker Taft, AdaCore, USA;
- "Introduction to the Alire Package Manager", by Quentin Dauprat, Université de Caen, Normandie, France.

Satellite Events

Three workshops are held on Friday 13th:

- 4th ADEPT workshop "AADL by its practitioners".

<http://www.ada-europe.org/conference2025/workshop_adept.html>

- 10th International Workshop on "Challenges and New Approaches for Dependable and Cyber-Physical System Engineering" (DeCPS 2025).

<http://www.ada-europe.org/conference2025/decps.html>

- 2nd "Ada Developers Workshop", an informal yet dynamic gathering for developers in the Ada community to meet, share insights, and present their latest projects or project updates; a full "Ada day" with 8 technical presentations on various Ada-related topics and 10 authors from 5 countries: Belgium, France, Spain, Switzerland, and USA.

<http://www.ada-europe.org/conference2025/workshop_adadev.html>

Social Program

<http://www.ada-europe.org/conference2025/social_program.html>

The conference provides several opportunities to socialize:

- Each day: coffee breaks and lunches offer ample time for interaction and networking with participants and vendors.
- Tuesday early evening: Welcome Reception (details to be announced).
- On conference days: Mineralogy Museum. The École des Mines features a splendid Mineralogy museum. During the conference, you can access it free of charge by showing your name badge. Do not miss this opportunity!
- Wednesday evening: Conference Banquet at the "Bouillon Racine", a typical Parisian restaurant built in 1906 and classified as "historical monument". The restaurant is located within walking distance from the conference location. We will have dinner in the beautiful dining room on the first floor, an authentic turn of the 20th century decoration.
- Thursday evening: Chill Event at the Café Latin, a casual typical café, to have a little chat with colleagues during an informal dinner (separate registration). For those who want to get a feeling of the Latin Quarter, we prepared an itinerary that will show you the main milestones of the neighborhood while walking to the Café Latin.

Further Information

Registration:

- registration information is provided at <http://www.ada-europe.org/conference2025/registration.html>
- early registration discount until Tuesday May 27, 2025

- payment possible by credit card or bank transfer
- special low conference fee for authors
- discount for Ada-Europe and ACM SIGPLAN members
- registration includes coffee breaks, lunches and social events
- strong discount on conference fees for students
- minimal fee for Ada Developers Workshop
- see registration page for all details

Promotion:

- recommended hashtags: #AEiC2025 #AdaEurope #AdaProgramming

The conference is organized by:

- Ada-Europe <http://www.ada-europe.org/>
- Ada-France <https://www.ada-france.org/>

in cooperation with:

- ACM SIGPLAN <http://www.sigplan.org/>

supported and sponsored by:

- Embedded France <https://www.embedded-france.org/>
- AdaCore <https://www.adacore.com/>
- Emenda <https://emenda.com/>
- Systerel <https://www.systerel.fr/>
- Scitools <https://scitools.com/>
- Pacte Novation <https://www.pactenovation.fr/index.php/logiciel-critique-ada-fondements-et-enjeux/>
- Institut Frederik Bull <https://institutbull.fr/>
- Mines Paris <https://www.minesparis.psl.eu/>

Please make sure you book accommodation as soon as possible.

<https://www.ada-europe.org/conference2025/accommodation.html>

For more info and latest updates see the conference website at <http://www.ada-europe.org/conference2025>.

We look forward to seeing you in Paris in June 2025!

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2025 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be
Dirk.Craeynest@kuleuven.be

* 29th Ada-Europe Int.Conf. Reliable Software Technologies (AEiC 2025)

(V5.1)

Ada-Belgium Spring 2025 Event, Sun 1 June 2025

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Belgium Spring 2025 Event, Sun 1 June 2025
Date: Mon, 19 May 2025 16:09:39 +0000
Newsgroups: comp.lang.ada

Ada-Belgium Spring 2025 Event

Sunday, June 1, 2025, 12:00-19:00

Leuven, Belgium

including at 15:00

2025 Ada-Belgium General Assembly

and at 16:00

Ada Round-Table Discussion

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/25/250601-abga.html

---

Announcement

The next Ada-Belgium event will take place on Sunday, June 1, 2025 in Leuven.

For the 14th time, Ada-Belgium organizes their "Spring Event", which starts at noon, runs until 7pm, and includes an informal lunch, the 32nd General Assembly of the organization, and a round-table discussion on Ada-related topics the participants would like to bring up.

Schedule

* 12:00 welcome and getting started (please be there!)
* 12:30 informal lunch
* 15:00 Ada-Belgium General Assembly
* 16:00 Ada round-table + informal discussions
* 19:00 end

Participation

Everyone interested (members and non-members alike) is welcome at any or all parts of this event.

For practical reasons registration is required. If you would like to attend, please send an email before Wednesday, May 28, 18:00, to Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be> with the subject "Ada-Belgium Spring 2025 Event", so you can get precise directions to the place of the meeting. Even if you already responded to the preliminary announcement, please reconfirm your participation ASAP.

If you are interested to join Ada-Belgium, please register by filling out the 2025 membership application form [1] and by paying the appropriate fee before the General Assembly. After payment you will receive a receipt from our treasurer and you are considered a member of the organization for the year 2025 with all member benefits [2]. Early enrollment ensures you receive the full Ada-Belgium membership benefits (including the Ada-Europe indirect membership benefits package).

As mentioned at earlier occasions, we have a limited stock of documentation sets and Ada related CD-ROMs that were distributed at previous events, as well as some back issues of the Ada User Journal [3]. These will be available on a first-come first-serve basis at the General Assembly for current and new members. (Please indicate in the above-mentioned registration e-mail that you're interested, so we can bring enough copies.)

[1] http://www.cs.kuleuven.be/~dirk/ada-belgium/forms/member-form25.html

[2] http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html

[3] http://www.ada-europe.org/auj/home/

Informal lunch

The organization will provide food and beverage to all Ada-Belgium members. Non-members who want to participate at the lunch are also welcome: they can choose to join the organization or pay the sum of 25 Euros per person to the Treasurer of the organization.

General Assembly

All Ada-Belgium members have a vote at the General Assembly, can add items to the agenda, and can be a candidate for a position on the Board [4]. See the separate official convocation for all details.

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

[5] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/25/250601-abga-conv.html

Ada Round-Table Discussion

As in recent years, we plan to keep the technical part of the Spring event informal as well. We will have a round-table discussion on Ada-related topics the participants would like to bring up. We invite everyone to briefly mention how they are using Ada in their work or non-work environment, and/or what kind of Ada-related activities they would like to embark on. We hope this might spark some concrete ideas for new activities and collaborations.

Directions

To permit this more interactive and social format, the event takes place at private premises in Leuven. As instructed above, please inform us by e-mail if you would like to attend, and we'll provide you precise directions to the place of the meeting. Obviously, the number of participants we can accommodate is not unlimited, so don't delay...

Looking forward to meet many of you!

Dirk Craeynest
President Ada-Belgium
Dirk.Craeynest@cs.kuleuven.be

Acknowledgements

We would like to thank our sponsors for their continued support of our activities: AdaCore, and KU Leuven (University of Leuven).

If you would also like to support Ada-Belgium, find out about the extra Ada-Belgium sponsorship benefits: http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html #sponsor

(V20250519.1)

Ada-related Resources

[Delta counts are interpolated* from April 1st to July 1st. Tabulated raw data is available at https://bit.ly/auj-signals —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 13 Oct 2025 23:36
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: 8_860** (-412) members [1]
- LinkedIn: 3_658 (+46) members [2]
- Stack Overflow: 2_443 (+5) questions [3]
- Ada-lang.io: 384 (+37) users [4]
- Gitter: 289 (+7) people [5]
- Telegram: 227 (+7) users [6]
- Discord: 192 (+1) users [7]
- Libera.Chat: 75 (=) concurrent users [8]

[*] This interpolation is intended to offer more representative and uniform delta counts from number to number of the Journal. The raw numbers at the actual date when they were taken can be found following the aforementioned link.

[**] Reddit has changed how it measures engagement. Rather than showing all subscribers since the Epoch, it now shows a running average of active users during the last month. Delta count is therefore not representative of any real changes in membership.

[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://discord.gg/fEmGe87c

[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: 13 Oct 2025 23:36
To: Ada User Journal readership

GitHub: >7_800* (=) repositories [1]
>1_000* (=) developers [1]

Alire: 1_426 (+64) releases [2]
559 (+23) crates [3]

Rosetta Code: 1_026 (+11) examples [4]
43 (+1) developers [5]

Sourceforge: 242 (=) projects [6]

Open Hub: 214 (=) projects [7]

Codelabs: 63 (+1) repositories [8]

Bitbucket: 37* (=) repositories [9]

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

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

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

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

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

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

[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: 13 Oct 2025 23:36
To: Ada User Journal readership

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

- TIOBE Index: 12 (+8) 1.65% (40.00%) [1]
- PYPL Index: 15 (=) 1.52% (=) [2]
- Stack Overflow Survey: 40 (=) 0.9% (=) [3]
- IEEE Spectrum (trending): 46 (=) Score: 0.0023 (=) [4]
- IEEE Spectrum (general): 50 (=) Score: 0.0015 (=) [4]
- IEEE Spectrum (jobs): 54 (+1) Score: 0.0001 (1.82%) [4]
- Languish Trends: 154 (-8) 0.01% (-5.48%) [5]

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

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

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

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

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

Birth of Wiki Site "Ada 83 Memory"

From: VMo
Subject: Birth of Wiki Site "ada 83 Memory" at Http://www.ada83.org/
Date: Sat, 5 Apr 2025 10:36:38 +0000
Newsgroups: forum.ada-lang.io

```
with TEXT_IO;
procedure BIRTH_NOTIFICATION is
begin
  TEXT_IO.PUT_LINE("Ada 83 Memory
   wiki is started at http://www.ada83.org/");
end;
```

/Jelle Hermsen/ and I (/Vincent Morin/) found useful to create a wiki dedicated to Ada 83 to collect stories, technical elements about the ANSI/MIL-STD-1815 language (compilers, programming environments…) and its machine supports as iAPX or Rational.

Vintage atmosphere from the two rich decades from 1975 to 1995 can be evoked, as well as today's still relevance of the language in 83 standard.

This is a call to veterans of the language to be witnesses of the Ada language of the origins.

Feel free to have a look at the site embryo at http://www.ada83.org/
your contributions will be welcome!

Looking forward to read your stories and reflections.

GitHub-hosted Ada Reference Manual

From: sttaft
Subject: Github-hosted Ada Reference Manual
Date: Tue, 8 Apr 2025 16:03:39 +0000
Newsgroups: forum.ada-lang.io

Some folks have had trouble reaching ada-auth.org [1] on occasion. There is now a "mirror" of some of the Ada Reference Manuals on a GitHub-hosted site:

https://ada-rapporteur-group.github.io/ [2]

It is very reliable and very fast, so probably the better place to use for daily access.

[1] http://ada-auth.org

[2] https://ada-rapporteur-group.github.io

Ada Badge

From: LionelDraghi
Subject: Is There an Ada Badge like the Alire One?
Date: Wed, 16 Apr 2025 14:43:14 +0000
Newsgroups: forum.ada-lang.io

It seems to me that there is no badge to display that we are proud to use Ada on our Git [Hub|Lab] pages, like there is for Alire.

I found a collection for other languages here: Badges4-README.md-Profile [1], but nothing with our logo!

(That could be based on the AdaCore excellent and flexible proposal (Our Contribution to the Ada Logo Discussion | The AdaCore Blog [2])).

Did I miss something? Any suggestion?

BTW, for the "Ada crate of the year" awarded projects, feel free to reuse this one [3] with the right title and link.

[1] https://github.com/alexandresanlim/Badges4-README.md-Profile?tab=readme-ov-file#-languages-

[2] https://blog.adacore.com/our-contribution-to-the-ada-logo-discussion

[3] https://blog.adacore.com/ada-spark-crate-of-the-year-2024-winners-announced

From: mgrojo
Date: Thu, 17 Apr 2025 11:10:41 +0000

Good idea.

According to Logos | Shields.io [1], to add a logo to the badge, it seems the first step is adding it to https://simpleicons.org/ following Sign in to GitHub · GitHub [2]

I've created the request as a first step, but maybe it doesn't get solved unless someone contributes a pull request.

For the moment, we can experiment with the inline image: https://img.shields.io/badge/inside-green?style=for-the-badge&logo=[...]

[Removed large binary base64-encoded data –arm.]

I don't know how to get a bigger logo.

[1] https://shields.io/docs/logos

[2] https://github.com/simple-icons/simple-icons/issues/new?assignees=&labels=new+icon&template=icon_request.yml

[3] https://github.com/simple-icons/simple-icons/issues/13071

[GSOC] Working on Open Source Ada Projects and Getting Paid?

From: Irvise
Subject: [gsoc & Ada] Working on Open Source Ada Projects and Getting Paid?
Date: Sun, 11 May 2025 18:52:47 +0000
Newsgroups: forum.ada-lang.io

I am very happy to announce that @sttaft proposal to add the Ada 2022 parallel keyword to the GNAT frontend has been approved and will be funded if all the milestones are met!

Google Summer of Code [1] is a global program focused on bringing more developers into open source software development.

Congratulations to the participants!

[1] https://summerofcode.withgoogle.com/programs/2025/projects/Ex0T0sK7

From: sttaft
Date: Sun, 11 May 2025 19:59:18 +0000

Yes, thanks. Ethan Luis McDonough will be doing the work, and Richard Wai and I will be his mentors. It will be great to have these parallel features of Ada 2022 available in the FSF version of GNAT.

Books by Prof. Burns and Prof. Wellings

From: ornithophile
Subject: Books by Prof. Burns and Prof. Wellings
Date: Thu, 15 May 2025 18:34:28 +0000
Newsgroups: forum.ada-lang.io

Hi all, experienced programmer here trying to build my Ada knowledge. I just bought /Concurrent and Real-Time Programming in Ada/ (3rd edition) and then noticed they have a newer book, /Analysable Real-Time Systems: Programmed in Ada/. Has anyone here read both? Is it worth it to get the new book as well? I'd imagine the older book might be missing some Ada 2012 bits.

From: sidisyom
Date: Sun, 18 May 2025 21:40:28 +0000

I've read both and found them excellent sources with numerous code examples (though not always fully compilable if I remember right) of concurrency patterns and other things like fully functional servers (Polling/Defferable).

I think the difference is mostly that the latter covers a lot more thoroughly some aspects of scheduling theory and actually the first part of the book deals exclusively with that whilst the second part provides many code examples which are often similar to the ones found in the first book. Also, like you mention, the first one is Ada 2005 whilst the second one is 2012, so there may be some variation in the syntax.

So yeah, if you want to have more code examples, I'd go with the first one initially and then maybe move on to the second one for a more thorough treatment of real-time programming.

Ada Developing Environment

From: ocyou
Subject: Ada Environment
Date: Mon, 19 May 2025 13:19:40 +0000
Newsgroups: forum.ada-lang.io

From my understanding, Ada is like Prolog in the sense that it's an ISO standard, with multiple implementations/compilers.

Where can I find the list of Ada implementations and their build tools, package managers, etc?

I'm using for now Alire, which I find straightforward in building/running/managing crates, etc. I tried GNAT, but it seems more complex (make files, etc)

I'm just looking for the most efficient learning path. I want tools that spare me the pain of all what happens under the hood.

Also, in terms of docs, I'm confused: should I check *adacore docs or ada-lang or ada-auth*..? (normally, they should be ISO-compliant, but I just want to confirm…)

From: OneWingedShark
Date: Mon, 19 May 2025 15:53:10 +0000

ARG is the Ada Rapportaur Group, which produces/develops the standard.

The standards made available on Ada-auth and on AdaIC *are* the ISO standards, minus some formatting/styling imposed by the ISO.

Ada-Auth is the ARG's site.

AdaIC is the Ada Information Clearinghouse.

So both of these are official Ada.

AdaCore is a company, not official Ada.

From: DirkCraeynest
Date: Tue, 20 May 2025 07:27:18 +0000

> The standards made available on Ada-auth and on AdaIC *are* the ISO standards, minus some formatting/styling imposed by the ISO.

So strictly speaking what is on the Ada-auth and AdaIC sites are not the ISO standards, but correspond to the sources from which the successive ISO standards were derived.

FYI, for those who like to have a hard copy of Ada RMs: all successive versions have been printed by Springer in their LNCS series. The consolidated Ada 2022 RM is currently being prepared (2022 RM + Corrigendum 1), once again an initiative of Ada-Europe.

From: jere
Date: Mon, 19 May 2025 15:39:43 +0000

Alire is a package manager that wraps around GNAT and a program called GPRBuild, so if you are using GNAT you have 3 separate ways to build, with varying levels of ease and customization:

- gnatmake command - very bare bones, command line with switches
- gprbuild command - a build system that uses a special *.gpr file to tell gnatmake how to build with all of your options/switches but in a much more readable way that can be customized much more
- alr build command - Alire specific command that wraps around gprbuild that uses special *.toml files to tell Alire how to manage gprbuild files and options but also adds even more customization options on top of what gprbuild allows. It also gives you access to a plethora of libraries and examples.

Note that gnatmake and gprbuild are both products of a specific company that makes GNAT called AdaCore. Alire is separately maintained by members of the community.

All of that is for one Ada compiler. There are others out there. Here is a list of some options:

[1] https://github.com/ohenley/awesome-ada?tab=readme-ov-file#compilers

From: pmnw
Date: Mon, 19 May 2025 16:02:11 +0000

Yes, Ada is a standardised language and has been implemented by multiple vendors.

Nowadays, only GNAT seems to implement the latest Ada specification. Beyond GNAT, there are few closed-source compilers that provide older versions of Ada.

If you're interested in what is still available, search for PTC or Janus/Ada or Greenhills. There's also the free and amicable HAC.

Alire is the leading toolchain wrapper. For example, it uses the GNAT compiler and the GPRbuild build tool.

There's also AURA although I haven't seen it used.

ocyou:

> I'm using for now Alire…

This happens to be the simplest way to develop in Ada right now and was recommended by AdaCore (the corporate weight behind Ada) after they phased out the community version of their IDE called GNATStudio.

Please also check out ada_language_server.

[...]

Nvidia Publishes SPARK Process to Meet ISO-26262 Requirements

From: Irvise
Subject: Nvidia Publishes Spark Process to Meet Iso-26262 Requirements
Date: Wed, 4 Jun 2025 20:55:02 +0000
Newsgroups: forum.ada-lang.io

NVIDIA and AdaCore [1] have jointly worked on publishing the process to get an Ada/SPARK autonomous driving system to be ISO-26262 compliant.

Best part is, *NVIDIA open sourced their documents* for everybody to see (see link above)! The page with the rendered documentation can be found here [2].

Also, I would like to bring your attention to the fact that this is not the only area NVIDIA is working with Ada/SPARK. Cybersecurity is also one of their main focus points [3].

I believe it is very important to communicate and showcase these results/choices. People think that Ada is not used anymore; if it is used, it is in legacy systems; if it is used in legacy systems it is being replaced, etc. Well, here is a fresh, fresh, fresh, out of the oven, ripe and ready to be digested bit of news coming from the currently most valuable company (depending on market conditions) and it is not related to AI.

It would be nice if some of you reposted this info in HackerNews/Reddit/Phoronix/etc and proactively and kindly answered people's questions about Ada, its usage, etc. Also, it is very important to note that SPARK, GNAT, GNAT-LLVM and SPARKlib are all open source programs/libs under rather liberal licenses and that people can just get a hold of them and use them right now!

[1] https://www.adacore.com/press/ada-and-spark-enter-the-automotive-iso-26262-market-with-nvidia

[2] https://nvidia.github.io/spark-process/

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

From: mgrojo
Date: Wed, 4 Jun 2025 21:47:46 +0000

Very good news. Til now, the fact that NVIDIA was using SPARK lacked a direct, undoubted reference from NVIDIA itself.

It's already on HackerNews.

[1] https://news.ycombinator.com/item?id=44184861

Ada-related Tools

Gprdeps: a New Tool for GPR Project Files Linting

From: ebriot
Subject: Gprdeps: a New Tool for Gpr Project Files Linting
Date: Tue, 1 Apr 2025 05:57:23 +0000
Newsgroups: forum.ada-lang.io

This tool is how I always thought gprbuild should have been implemented. I was part of the design discussions when AdaCore initially developed the GPR files, but we unfortunately chose to reuse the GNAT front-end for the parsing (for reasons still unclear to me, as the parser is really the simple part here). Having all dependency information in memory from the get-go, and not relying on external tools (the compiler) to discover the dependencies as we go opens the door for quite a number of things.

I remember we had an issue in the compiler at some point with the order of -l (libraries) on the linker command line. Having a graph of dependencies would have let us find a proper ordering with a simple topological sort for instance. One other idea is to easy add extra steps for generating sources (this is just going from source A to source B via code generation, then source B to object C via compilation for instance). Instead, gprbuild uses ad-hoc algorithm every time.

I have written the tool in Rust because I wanted to learn that language. It would just as easily have been written in Ada I am sure. I must say I was very surprised by the speed of the executable, which I attribute to compiler optimizations that are possible because of the lack of aliasing. But Ada would likely have been pretty close too.

Parallel For-loops in Ada

From: leo-brewin
Subject: Parallel For-loops in Ada
Date: Tue, 1 Apr 2025 08:27:15 +0000
Newsgroups: forum.ada-lang.io

I've created a GitHub repository describing my experiments in using Ada's tasking tools to run for-loops across multiple cpu cores. There are a handful of example codes using rendezvous calls and protected objects. I'm posting the link here as it might be useful for others wanting to play this game.

You can find the repository at github.com/leo-brewin/ada-parallel-for-loops [1].

You might also like to look at LWT, the Light Weight Threading library, from the ParaSail project. This is a much more serious attempt at this game than my little effort.

The LWT sources can be found here [2].

See also the announcement of LWT on the Ada-forum here:

Lightweight Parallelism library based on Ada 2022 features [3].

A copy of the LWT sources can also be found in my repo (under examples/lwt).

[1] https://github.com/leo-brewin/ada-parallel-for-loops

[2] https://github.com/parasail-lang/parasail/tree/main/lwt#light-weight-threading-library-for-ada-2022

[3] https://forum.ada-lang.io/t/lightweight-parallelism-library-based-on-ada-2022-features/516

From: sttaft
Date: Tue, 1 Apr 2025 12:47:36 +0000

Thanks for posting this! It's great to see this sort of use of parallelism.

Also thanks for the nice references to the LWT library. One goal of the LWT library is to avoid creating "heavy weight" tasks more frequently than necessary, but instead reusing them. Your timings show that creating and finalizing tasks is probably not a significant time sink, so long as you do enough work with them. The other goal of the LWT library was to support the parallel execution of a heterogeneous collection of light-weight threads. It sounds like that is not too important in your application area, since you are mainly focused on parallel loops, rather than something that might need a more general divide-and-conquer approach.

From: Irvise
Date: Tue, 1 Apr 2025 20:40:27 +0000

THANK YOU!

I have always been missing some benchmarks on Ada's tasking system and its (performance) behaviour! And also how it compares to other common parallelism models. LWT, afaik, uses OpenMP behind the scenes, which to me, is the parallelism standard (for a single CPU chip). It is great to see that tasks perform quite well! So thank you very very much for the tests

P.S.: as a LaTeX aficionado and I see you also use Cadabra, you just got a new follower.

From: sttaft
Date: Thu, 10 Apr 2025 01:20:56 +0000

Actually, LWT uses either OpenMP or a home-grown work-stealing scheduler (which is often faster than OpenMP). LWT has a "plug-in" architecture so you can insert other light-weight-thread schedulers underneath, by "with"ing the package that defines the scheduler. If you don't plug in any of them, LWT just runs the threads sequentially.

Archiving of GNATdashboard on GitHub

From: ch13
Subject: Question about the Archiving of the Gnatdashboard Project on Github
Date: Tue, 1 Apr 2025 16:15:23 +0000
Newsgroups: forum.ada-lang.io

I noticed that the gnatdashboard project [1] has been archived on GitHub since version 23 in April 2023. However, in my company, we continue to use version 25 of this product.

My question is: was this archiving in April 2023 due to a change in AdaCore's strategy? What are the reasons behind this decision?

Thank you in advance for your responses.

[1] https://github.com/AdaCore/gnatdashboard

From: Irvise
Date: Tue, 1 Apr 2025 20:28:32 +0000

Hi @ch13 and welcome to the forums!

@Fabien.C can probably provide a much nicer answer than I will, but still, I'll try.

From what I have talked with AdaCore people, they are focusing the open source efforts to what mostly benefits the community and does not interfere with their commercial offering (creating unwarranted expectations for example). From what I know, they have stopped releasing open updates to some products that are mainly only useful for commercial customers and products they would like to phase off in favour of others. Since your company probably has a GNATPro license, they will still get updates on the products, even if they are not publicly released.

I hope this helps!

P.S.: in case some people think AdaCore does not care about open source, read their blog and see how they changed their licenses from the GPL family to Apache and how their best products, such as GNAT and SPARK are fully open. Also, we the community get the features first, and no, that is not because we are used as guinea pigs. The new updates tend to be quite robust from the very beginning

From: ch13
Date: Wed, 2 Apr 2025 07:37:09 +0000

Hi @Irvise,

Thank you for your response. When you say that a product could be phased out in favor of others, do you mean that some tools will gradually be used less?

Currently, I am doing code quality analysis using GNAThub via the CodePeer plugin, which runs GNATSAS. Do you think this is still relevant? Or in the future, will we have to use GNATSAS directly or another tool?

That's a lot of questions

Thank you in advance.

From: Irvise
Date: Wed, 2 Apr 2025 21:34:47 +0000

An example of product transition is GNATpp (pretty print) being phased out in favour of GNATformat.

Regarding the other products you mention, they are for customers. I am not an AdaCore customer, so I cannot comment on them… Nonetheless, AdaCore makes their product line roadmap public, so you may find some more information there: https://docs.adacore.com/live/wave/roadmap/html/roadmap/index.html

LEA 0.92

From: zertovitch
Subject: Lea Version 0.92
Date: Wed, 2 Apr 2025 02:28:28 +0000
Newsgroups: forum.ada-lang.io

LEA is a Lightweight Editor for Ada

Alire crate: https://alire.ada.dev/crates/lea

Source repository #1: https://github.com/zertovitch/lea

Source repository #2: https://sf.net/p/l-e-a/code/HEAD/tree/

Web site: http://l-e-a.sf.net/

Changes since last announcement here:

- Option box for HAC: you can choose the warnings & notes level
- Recognizes source path comments in HAC programs (source path is appended during build)
- Added 5 HAC examples (available through menu: Action / Code sample)

The LEA executable is standalone, holds in 4 MB and doesn't require an installation procedure.

Features:

- Multi-document
- Multiple undo's & redo's
- Multi-line & multi-point edit, rectangular selections
- Color themes, easy to switch
- Duplication of lines and selections
- Syntax highlighting
- Parenthesis matching
- Bookmarks
- Includes HAC, the HAC Ada Compiler
- Smart editor features (auto-completion, navigation, mouse-hover infos)
- Includes numerous examples of Ada programs, ready to be run
- Single executable, runs without installation
- Free, Open-Source

LEA is programmed in Ada, as well as HAC.

AZip 2.7.1

From: zertovitch
Subject: Azip Version 2.7.1
Date: Sat, 5 Apr 2025 05:46:28 +0000
Newsgroups: forum.ada-lang.io

AZip is a Zip archive manager.

Alire crate:
https://alire.ada.dev/crates/azip

Project site 1:
https://github.com/zertovitch/azip

Project site 2:
http://sourceforge.net/projects/azip

Home page: http://azip.sf.net/

Latest changes:

- Added option to make a backup for Update or Recompression operations
- Added brute-force Recompression mode

Some features of AZip:

- Aimed at simplicity, minimal option set
- Usual archive management: Add / Remove / Extract files
- Original, useful built-in tools:
- Text & name search function through an archive (no file is written during search)
- Archive updater
- Archive recompression, using brute force or an algorithm-picking approach for improving a Zip archive's compression.
- Flat view / Tree view toggle
- Multi-document
- Encryption
- Zip compression formats supported: Reduce, Shrink, Implode, Deflate, Deflate64, BZip2, LZMA
- Portable (no installation needed, no DLL); can operate with a config file instead of the registry (stealth mode)
- Installable if desired (application is its own installer)
- Free, open-source

The AZip executable is standalone, holds in 3 MB and doesn't require an installation procedure.

However, you can install AZip: the executable is its own installer and will offer to install itself.

"Under the hood" features:

- /AZip/ is from /A/ to /Z/ programmed in Ada
- Uses the highly portable Zip-Ada library - all in Ada.
- (regarding Windows "skin") Uses the GWindows library - all in Ada.

AdaStudio 2025.2 Free Edition

From: AdaStudio
Subject: Announce: Adastudio-2025.2 Release 02/04/2025 Free Edition
Date: Thu, 10 Apr 2025 10:32:03 +0000
Newsgroups: forum.ada-lang.io

AdaStudio-2025.2 release 02/04/2025 free edition based on Qt-6.9.0-everywhere opensource (expanded with modules from Qt-5.15: qtgamepad, qtx11extras, qtwinextras), VTK-9.4.1, FFMPEG-7.1, OpenCV-4.11.0, Whisper-1.7.4, SDL2-2.24.0, QtAV-1.13 MDK-SDK(wang-bin)

Qt6ada version 6.9.0 open source and qt6base.dll, qt6ext.dll (win64), libqt6base.so, libqt6txt.so(x86-64) built with Microsoft Visual Studio 2023 x64 Windows, GCC amd64 in Linux.

Package tested with GNAT gpl 2020 Ada compiler in Windows 64bit, Linux amd64 Debian 12.2.

AdaStudio-2025.2 includes next modules: qt6ada, vtkada, qt6mdkada, qt6cvada (face recognition, face detection, face identification, objects detection, QRcode detector, BARcode detection and others) and voice recognizer.

Qt6Ada is built under GNU LGPLv3 license.

Qt6Ada modules for Windows, Linux (Unix) are available from:
https://drive.google.com/drive/folders/0B2QuZLoe-yiPbmNQRl83M1dTRVE?resourcekey=0-b-M35gZhynB6-LOQww33Tg&usp=sharing

WebPage is
https://r3fowwcolhrzycn2yzlzzw-on.drv.tw/AdaStudio/adastudio.html

GCC v15.1 Ada Changelog

From: Irvise
Subject: Gcc v15.1 Ada Changelog
Date: Fri, 11 Apr 2025 12:55:37 +0000
Newsgroups: forum.ada-lang.io

GCC 15.1 should be released in the next few weeks once all the main bugs have been squashed. Nonetheless, the changes that have taken place for GNAT are already available in the Changelog page for GCC. There are quite a few nice things that I am personally very excited about!

https://gcc.gnu.org/gcc-15/changes.html

From: Lucretia
Date: Fri, 25 Apr 2025 13:18:49 +0000

Interesting…they added gcobol.

SweetAda Implements Some of Its Libc in Ada

From: Irvise
Subject: Sweetada Just Started to Implement Some of Its Libc in Ada
Date: Sun, 13 Apr 2025 20:31:14 +0000
Newsgroups: forum.ada-lang.io

I was informed recently that the SweetAda project [1], whose sole author is @SweetAda, has started to port some of its homegrown LibC implementation to Ada! You can find the c_wrapper.ad[s | b] files in the source tree and take a look at them [2]!

Remember, if you would like to help, you can always contribute by testing the code, making patches and fixes, writing documentation or supporting the authors!

My personal congrats to Gabriele! Best regards,

[1] https://github.com/gabriele-galeotti/SweetAda

[2] https://github.com/gabriele-galeotti/SweetAda/blob/d0d9da3605e269df817d7ded415422c190769648/clibrary/c_wrappers.ads

Ada Bootstrap Compiler Funding Thanks to Nlnet!

From: Irvise
Subject: Ada Bootstrap Compiler Funding Thanks to Nlnet!
Date: Wed, 23 Apr 2025 16:09:02 +0000
Newsgroups: forum.ada-lang.io

The nlnet foundation has announced the funding grant for an Ada Bootstrap compiler [1]!

I am a bit surprised by this news as I was involved in some behind the scenes conversations about it but I was not expecting it to pop up as soon as this! I will try to gather more information and see if there is a candidate selected or if it is open to the public to select a candidate.

[1] https://nlnet.nl/news/2025/20250422-announcement-grants-CommonsFund.html

From: OneWingedShark
Date: Wed, 23 Apr 2025 18:48:56 +0000

> Ada is an important computer language with a long history, with the compilers being built for new architectures in an ad-hoc basis based on previously existing Ada compilers from other architectures. *This project aims to create a bootstrap path from the C language to an Ada compiler without relying on an existing Ada compiler binary.* This will allow us to have a fully auditable trail from C to a working Ada compiler, removing concerns about hidden backdoors or other issues that may arise from using a compiler without a clear bootstrap path.

I have three reactions:

- A bootstrap implementation, yay!
- Not using existing tech to leverage the bootstrap process; disappointing, but understandable.
- Using C… **sigh** (though it is auditable, yay-ish… that should be a self-hosted feature.)

IMO, using self-hosted SPARK-verified Ada (employing Annex H), to self-host the compiler, having a modular backend/codegen would be a better route: aside from the compiler, create two subprojects: (1) a SeedForth [small-wordset, tokenized Forth] interpreter, SPARK verified; and (2) a SPARK-verified SeedForth emitting backend. — You can then feed things into each other so that you get the ability to interpret Ada on a new architecture implementing about 30 Forth-words, with provably correct interpreter and codegen. Native-code generation then becomes implementing a new backend, and you can use the previous results to generate a native-code compiler.

From: Irvise
Date: Sat, 26 Apr 2025 14:13:48 +0000

I will try to shed some light on some of the topics raised.

> Not using existing tech to leverage the bootstrap process; disappointing, but understandable.

It is using the available tech… in the constrained environment in which this is taking place

The thing is, this is no normal bootstrap compiler, where one can pick and use any available technology to carry such a task. The framework for this project is the live-bootstrap [1] system. This is a project that is trying as hard as possible to create a root-of-trust from where the world can be rebuilt in a bit-by-bit reproducible and transparent manner, in a way that a human being could verify the process and each and every single line of code. And this is done with the absolute bare minimum, which means that they are starting from a tiny binary seed to bootstrap all the way to modern Linux, GCC and Guile.

> Using C… /*sigh*/ (though it is auditable, yay-ish… that should be a self-hosted feature.)

C and Scheme are some of the first languages bootstrapped in the current live-bootstrap system, C being the more complete and capable of the set (the Scheme they implemented is incredibly small and it is only used to create a basic LibC and barebones C compiler). We would like to have Ada bootstrapped as early as possible, so C is the ideal system. However, other ideas have been tossed around, such as Go, a more advanced Scheme (TR7 or Guile), Python or Rust for example.

[...]

[1] https://github.com/fosslinux/live-bootstrap/blob/2057d551e0d072f85dd3c8b046e90e6be81a3604/parts.rst

Open Source Ada 83 Compilers

From: VMo
Subject: Discussions about Open Source Ada Compilers
Date: Sat, 10 May 2025 12:04:38 +0000
Newsgroups: forum.ada-lang.io

Rather than diverging on the thread on GNAT bootstrapping nlnet project when answering OneWingedShark message, I prefer opening an appropriate new thread.

[...]

From: VMo
Date: Sat, 10 May 2025 15:20:01 +0000

More directly in line with the topic title:

What do we have at hand in relation with Ada open source compiler and what can we do which would be a useful community achievement.

First, I set GNAT apart. GNAT is a wonderfully useful tool, up to date with recent Ada revisions, its source code can be looked at, but it is only manageable with a team of first class engineers. AdaCore does an admirable job, this is not our play court.

Briefly, what are the other compilers whose source code is available, what are their characteristics what can we do with them.

Ada/Ed

Ada/Ed as everybody knows is an early demonstrator Ada 83 compiler written in C, targets an interpreter and lacks very useful features as record representation clauses which are basic to microcontroller Ada programming (for example the AVR map is conveniently represented with such a record).

Working with Ada/Ed source code is a pain for non-C programmers. The high-level representation of Ada/Ed is not DIANA (which is not that easy to use, but is relatively clear, documented and used both in the Ada 83 Peregrine source code and the Pascal IIPS module of M.Ciernak). Ada/Ed can surely be recompiled with modern C compilers with some modest editing. I have done it for the interpreter being curious of its byte code and runtime.

But I doubt that Ada/Ed will be really useful outside of its initial vocation which was research and education being the SETL specification descendant.

SmallAda and its descendant HAC

Gautier de Montmollin has done an admirable job over a decade and intensely over the five last years, when recovering the SmallAda system written in Pascal. HAC is an operational system able to execute some serious programs, chapeau for this work. Nonetheless some choices have been made which will hinder or preclude some desirable features. HAC is not written in Ada 83, but at least Ada 95 using child packages. I don't know enough of the HAC source to tell if some Ada 2X features are used. On the other hand, HAC compiles a subset of Ada 83, SmallAda was a vehicle for tasking research and not destined to be a full Ada 83 compiler.

That means that HAC cannot compile itself, and should be augmented as far as being able to compile its Ada 9X or later 2X if necessary. This is probably not the goal of the author.

The Peregrine Ada 83 / DIANA and its descendant (TLALOC ?)

From the Walnut distribution, I extracted an Ada 83 system written for DEC Ada by Peregrine Systems directed by Bill Easton. As it was somewhat strangely structured (there was an unclear mix of three usage of IDL for IDL itself, LALR and DIANA management) I did my best to clarify the affair and modified the system's structure, virtual pages pointer management and many other things. Thanks to GNAT, that must be said.

This system takes full Ada 83 source text and produces a DIANA high level intermediate representation of it.

As far as I tested it, it works well but for some rudimentary library management and care to manually compile modules in the correct order.

This system also swallowed the Kalinda OS code, and if somebody tries other Ada 83 source text (which should be done), I am confident it will produce the corresponding DIANA.

Then I decided to follow the polish IIPS project and began to browse the DIANA IR in the code_gen procedure to produce low level IR and final code. I think I'll call this whole system TLALOC.

Being written itself in rigorous Ada 83, TLALOC compiles itself without error and so produces all the DIANA structure of the compiler, which is already something. If a successful DIANA to low level IR code generator can be produced, this compilation system can become the only Ada 83 open source compiler which compiles itself (Ada/Ed cannot, being in C, HAC either being in 9X).

Ada 83 being a fixed standard, there is minimal risk of sideslipping from version to version as happens in GNAT. Only avoid inventing pragmas… Stick to the 83-LRM.

I rewrote the readme and markdown introductions in English on the Framagit with some clickable diagrams to access source code. Hoping it will encourage reading.

recall, the thing is here: Vincent Morin / Ada-83-compiler-tools · GitLab [1]

Is there anything else?

[1] https://framagit.org/VMo/ada-83-compiler-tools

From: sttaft
Date: Sat, 10 May 2025 18:37:31 +0000

The open-source ParaSail interpreter/compiler (all written in Ada) has a partial Ada 2022 front end. See parasail/ada202x_parser at [1] for the parser/lexer.

Internal tree representation, and static and dynamic analysis phases, are all in parasail/semantics at [2]

Interpreter/run-time is in parasail/interpreter at [3]

Translator from ParaSail VM (PSVM) to LLVM is in parasail/lib [4] in (ParaSail) source files such as compiler.{psi,psl}

[1] https://github.com/parasail-lang/parasail/tree/main/ada202x_parser

[2] https://github.com/parasail-lang/parasail/tree/main/semantics

[3] https://github.com/parasail-lang/parasail/tree/main/interpreter

[4] https://github.com/parasail-lang/parasail/tree/main/lib

From: waleedmebane
Date: Sun, 11 May 2025 02:47:47 +0000

Wow, that's impressive! (I read parasail_intro.pdf.) I wasn't aware of ParaSail as a new programming language though I've seen you mention ParaSail in posts in the context of OpenMP support and parallel for loops.

From: VMo
Date: Sun, 11 May 2025 07:19:23 +0000

Well, extremely interesting but somewhat stunning, I have no voice when considering the amount of work… The GitHub mentions only 3 contributors, I suppose there is some team behind. It is the work of how many people?

[...]

From: sttaft
Date: Sun, 11 May 2025 12:15:59 +0000

If you want to read most of the story of how ParaSail was designed, I would encourage you to take a look at the Blog which was started in 2009 and traced the week-by-week progress (Designing ParaSail, a new programming language: Why design a new programming language? [1]).

Having spent most of my career involved with the design and implementation of Ada, in 2009 I decided to start from scratch and design a stripped-down language focused on supporting multicore hardware, with high integrity. There was no "team" behind ParaSail, though it is based on things that were learned from collaborative work done over 40 years.

[1] https://parasail-programming-language.blogspot.com/2009/09/why-design-new-programming-language.html

From: VMo
Date: Mon, 12 May 2025 08:58:13 +0000

Thank you! I read the blog integrally. It is full of very interesting thoughts. Parasail is a leading edge research vehicle, somewhat far from my Ada 83 preservation and revival occupations.

But the modern and the ancient have some continuity.

For example I wondered why Ada 83 appeared to me to be such a peculiar software development when I was a young student and researcher; and why Ada 95 conveyed a more conventional feeling. I realized later that Ada 83 is a structure forcing language (to the point of being sometimes heavy on the software developer) and in fact discourages the use of pointers; of course there are access types, but much can be done without. On the contrary Ada 95 led the programmer to instinctively use aliased and access things and was much more conducive to pointers use. I'll say a horrible thing, but Ada 95 had a kind of C facility for the developer. So the "no pointer" policy of ParaSail echoes the "pointer set aside" encouragement of Ada 83.

Also, the "growing/shrinking" objects idea echoes the Ada 83 memory manager of my Kalinda OS trials. Kalinda's memory package is an Ada 83 (and at first Ada 95, after Pascal and shortly C, nobody's perfect) adaptation of the Macintosh memory manager and its double indirection handles. Ada forced me to implement a concept of dynamic size array with header to replace traditional pointer access to handles' contents. Pointers disappear, arrays are used. The same idea appears in ParaSail.

So thank you for all those intellectually vibrant ideas, and hoping good wind for ParaSail! (we are on the shore here in Brest, sailing evocates familiar pictures)

From: LionelDraghi
Date: Sun, 11 May 2025 20:05:12 +0000

This weekend, while clearing out the house where I grew up, I came across my papers from my final year at university. I think it was in 1989. I found articles and general documentation.

For example, the documentation for the Meridian compiler that we used at the university. Articles about the Alsys compiler (which I later used extensively in the industry), or about the amazing Rational system.

These Ada 83 systems were good and mature.

But they no longer exist, and haven't for years (even decades), and there's probably no more economic interest behind them.

I don't know who owns those software, but maybe we could ask for the current owners to release the sources under a free software licence, and give those marvelous pieces of code a second life. It would be a much easier starting point.

I regularly use FreeCAD. Its geometric kernel comes from Euclid, a software acquired by its competitor CATIA, and whose development ceased in 1998. Then, after some twists and turns, it was "liberated" and is now used in many open-source software and by many research projects.

Open Cascade Technology (OCCT, formerly named CAS.CADE) is an object-oriented C++ class library for 3D computer-aided design (CAD), computer-aided manufacturing (CAM), computer-aided engineering (CAE), etc. It is developed and supported by Open Cascade SAS company. It is free and open-source software released under the GNU Lesser General Public License (LGPL), version 2.1 only, which permits open source and proprietary uses. OCCT is a full-scale boundary representation (B-rep) modeling toolkit...

It would be great if the Alsys compiler could experience a similar resurrection.

[1] https://en.wikipedia.org/wiki/Open_Cascade_Technology

From: VMo
Date: Mon, 12 May 2025 08:22:11 +0000

This is obviously a good idea. The difficulty will be to find contacts, time has already passed, some people are retired or even passed away, the history of firms like Alsys is complicated between Thomson, Aonix, PCT, Atego. Are there any Ada 83 compiler sources left? Where? Who could have a hand on them?

If some of us have relations or contact indications, it would be welcome.

I contacted RRSoftware asking if they would be interested in participating in an Ada 83 technology preservation action. We'll see if there is an answer.

I also contacted the Karlsruhe university (KIT) Institut für Programstrukturen und Datenorganisation (IPD), I remember some articles from Rudolf Landwehr who is at Atos and affiliated to KIT. They have done very interesting work, could it be that some source be disclosed?

Let's try…

From: zertovitch
Date: Mon, 12 May 2025 09:18:40 +0000

> It would be great if the Alsys compiler could experience a similar resurrection.

The Alsys compiler is not dead, it changed its name to ObjectAda and is doing well.

You will notice .prj files in some open-source projects, including mine, which are project files for the ObjectAda IDE.

From: LionelDraghi
Date: Mon, 12 May 2025 10:21:29 +0000

As far as I remember, when switching to Ada 95, Alsys bought the technology, and dropped the in house technology.

I remember the ObjectAda interface being very different from the old Adaworld interface. I'm not sure if there is a single line of code shared between the two families (maybe for some backend?).

This is why I think the Ada83 code could be released.

But I may be wrong!

From: VMo
Date: Mon, 12 May 2025 11:15:42 +0000

I should have been more precise in the topic title and state that the interest is on open source *pure Ada 83* compilers.

Ada 9X and 2X is still sensitive technology (so more as we get close to our epoch) and having access to those compiler source code is improbable. It should not be the case for pure Ada 83 which is normally considered as obsolete technology.

If there was an Alsys pure Ada 83 compiler, it would be interesting to have access to its source code. I always used Object Ada in the past, and never used a pure Ada 83 Alsys compiler.

Anyway, I am convinced that it is time and perhaps urgent to preserve Ada 83 technology artifacts both for computer history and give the occasion to the young to have a hand on rather remarkable software engineering achievements.

I am reading the documents on the Rational R1000s400 restarting at DataMuseum.dk. They have produced an emulator starting from schematics… It is extremely interesting, and from some points of view still modern. Reading the patents is also instructive even in 2025.

For now I see three possible paths to Ada 83 compiler source code:

- Janus Ada 83 which was for i386 DOS and Unix
- Karlsruhe compiler which probably targeted a Siemens machine and M68K (to verify)
- and Alsys (targets?).

If somebody has other ideas or personal contacts, feel free to participate in the context of *Ada 83 Memory* wiki (ada83.org [1]).

The link [2] has a section "ancient".

It mentions Alsys AdaWorld, Verdix VADS, Meridian.

That is where we should try to convince some people to disclose and save old Ada 83 only compilers. There should not be any technological or commercial risk for them to do so.

[1] http://ada83.org

[2] https://www.adaforge.org/DevTools/DevToolsBuild/Compilers/

CoAP-SPARK 0.9.0

From: mgrojo
Subject: Coap-spark 0.9.0 Released
Date: Tue, 20 May 2025 21:31:28 +0000
Newsgroups: forum.ada-lang.io

CoAP-SPARK [1] is a library implementing the Constrained Application Protocol (CoAP) as defined in RFC 7252 [2], developed in SPARK and verified to be free of run-time errors (silver level).

This version implements the client side of the protocol with some limitations:

- It does not support block-wise transfers (extension to RFC 7252).
- It does not support retransmission of messages.
- It only supports NoSec and PreSharedKey security modes.

An example and tool implementing a command-line CoAP client can be found in the coap_client [3] crate.

[1] https://alire.ada.dev/crates/coap_spark

[2] https://www.rfc-editor.org/rfc/rfc7252

[3] https://alire.ada.dev/crates/coap_client

From: mgrojo
Date: Tue, 20 May 2025 21:39:18 +0000

Documentation of the project (my Master's Thesis draft) can be found in https://github.com/mgrojo/coap_spark/releases/download/v0.9.0/Memoria_TFM_CoAP-SPARK-Spanish.pdf (sorry, but only in Spanish). Any comment or feedback is appreciated.

Globe_3D 2025-04-18

From: zertovitch
Subject: Globe_3D 2025-04-18
Date: Thu, 22 May 2025 01:26:29 +0000
Newsgroups: forum.ada-lang.io

GLOBE_3D is a real-time GL Object Based 3D Engine written in Ada.

Alire crate:
https://alire.ada.dev/crates/globe_3d

Project site 1:
https://github.com/zertovitch/globe-3d/

Project site 2:
https://sourceforge.net/projects/globe3d/

Home page:
https://globe3d.sourceforge.io/

Latest changes:

- Code cleanup
- Visualisation of portals (on show_portals = True)

Ironclad Kernel 0.7.0

From: dragon-spirit-wtp
Subject: New 0.7.0 Release of Ironclad Kernel
Date: Sat, 31 May 2025 06:05:33 +0000
Newsgroups: forum.ada-lang.io

https://codeberg.org/Ironclad/Ironclad/releases/tag/v0.7.0

Non-breaking changes

- Greatly improved block device (SATA, NVMe) caching. Leading to big performance improvements.
- Added a basic NVMe driver.
- Implement support for x86's SMAP, and expose it in an architecture-independent way.
- Harden...

GNAT FSF 15

From: csagaert
Subject: Gnat Fsf 15 Released
Date: Thu, 5 Jun 2025 09:13:55 +0000
Newsgroups: forum.ada-lang.io

We released GNAT FSF 15 on Alire yesterday. This release is based on GCC 15.1.0; the release notes for Ada can be found here [1].

There are prebuilt binaries for Linux (x86_64 and aarch64), Windows (x86_64) and macOS (x86_64 and aarch64). Please note that the binaries for macOS aarch64 are based on an unofficial branch of GCC that has not yet released a final version for GCC 15, so there may be some issues that are absent from other platforms.

Cross compilers are also available on all platforms for the following targets: arm-elf, riscv64-elf, avr-elf and xtensa-esp32-elf.

You can report issues with the released compilers on this repository [2].

We want to thank @simonjwright for his work on building GNAT for macOS. His work helped us greatly to provide functional compilers for the platform. He will be sorely missed.

[1] https://gcc.gnu.org/gcc-15/changes.html#ada

[2] https://github.com/alire-project/GNAT-FSF-builds

Filoject, a Dependency Injection Framework for Ada

From: rehartmann
Subject: Filoject, a Dependency Injection Framework for Ada
Date: Sun, 22 Jun 2025 10:32:24 +0000
Newsgroups: forum.ada-lang.io

Filoject [1], a dependency injection framework for Ada, is now available.

What is this good for, you may ask?

I'm not claiming that Ada needs a DI framework, nor that it may even be a good idea to use one in Ada projects.

But given that practically all programming languages competing with Ada have DI frameworks, this may be something which is worth experimenting

with. Filoject is meant to provide a basis for this experimentation.

Filoject uses a code generator, a custom pragma and a custom aspect.

[1] https://github.com/rehartmann/filoject

Simple Components 4.74

From: dmitry-kazakov
Subject: Simple Components 4.74
Date: Mon, 30 Jun 2025 10:48:45 +0000
Newsgroups: forum.ada-lang.io

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), arbitrary precision arithmetic, pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, streams, persistent storage, multiple connections server/client designing tools and protocols implementations.

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

The focus of this release is an implementation of arbitrary precision rational numbers. The difference to standard library Ada.Numerics.Big_Numbers.Big_Reals:

- No limits, except for the pool size;
- String edit packages representing a number in a usual form (with the specified accuracy) rather than as a numerator/denominator ratio;
- An equivalent of Ada.Numerics.Elementary_Functions with rational approximations;
- Simple continued fractions support.

Changes to the previous version:

- The package Unbounded_Rationals provides an implementation of arbitrary precision rational numbers arithmetic;
- The package Unbounded_Rationals.Elementary_Functions provides approximations in rational numbers of some elementary, trigonometric and hyperbolic functions;
- The package Strings_Edit.Unbounded_Rational_Edit provides string editing for arbitrary precision rational numbers;
- The package Strings_Edit.Unbounded_Unsigned_Edit was optimized for better performance;
- Mixed Unsigned_Integer to Integer operations were added to the package Unsigned_Integers;
- The package Unbounded_Rationals.Continued_Fractions provides an implementation of simple continued fractions;
- The package Strings_Edit.Continued_Fraction_Edit provides string editing for simple continued fractions;
- Bug fixes in List_SetItem Python bindings;
- Check_Error message in Python binding was improved, trace back output added;
- Added higher level operations to create Python modules after initialization of the Python interpreter;
- Added higher level operations to create Python classes AKA heap types after initialization;
- An example of higher level module creation was provided;
- An example of higher level class creation was provided;
- IsInitialized function was added to Python bindings;
- OSX project settings fixed.

Ada-related Products

ObjectAda 10.6

From: JC001
Subject: Objectada 10.6 Released
Date: Wed, 21 May 2025 16:49:03 +0000
Newsgroups: forum.ada-lang.io

PTC announced the release of ObjectAda 10.6 [1] on 2025-03-03.

Those with a documented history (such as on-line repositories) of developing Ada open-source projects may obtain a free copy of ObjectAda by writing directly to Shawn Fanning, Software Development Director, user name sfanning at the domain ptc.com.

[1] https://ptc-p-001.sitecorecontenthub.cloud/api/public/content/f00b267f47f8426c9e9e41b01d5c4bbf

References to Publications

Article: Nvidia Drives Ada and SPARK into Driverless Cars

From: dragon-spirit-wtp
Subject: Article: Nvidia Drives Ada and Spark into Driverless Cars
Date: Fri, 6 Jun 2025 06:50:49 +0000
Newsgroups: forum.ada-lang.io

This is a great article that highlights various virtues of developing with Ada and SPARK that contributed to NVIDIA's recent achievement of certifying their DriveOS automotive operating system to the /highest/ automotive level of safety, ASIL-D. The first ever!

eeNews Europe – 4 Jun 25 [1]

Nvidia drives Ada and SPARK into driverless cars [1]

AdaCore and Nvidia have developed an open source reference flow for the Ada and SPARK formal languages in driverless cars.

Est. reading time: 5 minutes

Here is a paragraph from the article:

"Adopting a new programming language involves deploying a new environment, training teams to a new formalism, adapting programming patterns and many other issues. However, from a process standpoint, programming languages are vastly interchangeable, but Ada and SPARK is a different story."

[1] https://www.eenewseurope.com/en/nvidia-drives-ada-and-spark-into-driverless-cars/

Ada and Other Languages

Unexpected Ada Advocacy

From: LionelDraghi
Subject: Unexpected Ada Advocacy
Date: Tue, 1 Apr 2025 20:22:30 +0000
Newsgroups: forum.ada-lang.io

I was watching a video of a PyCon 2018 conference talk by Hillel Wayne titled Beyond Unit Tests: Taking Your Testing to the Next Level [1]

He was starting a "Design by contract" part of the speech (about min 18), and said:

> It turns out this is actually one of the most powerful verification techniques we know about.
> We've seen it work empirically at scale in all sorts of places but these places have mostly been avionics critical systems defense academia so nobody's heard of this thing which honestly I find completely crazy.
> I mean here's a question how many of you've heard of Ada?
> How many of you actually looked into what Ada can do?
> OK I've actually done some experimentation: you can do incredible things with Ada, but because it's Ada nobody cares.

Couldn't summarize the situation any better

And then later (around min 28), about Formal Method:

> This is the study of how we can prove software is correct at a level not seen in any kind of modern-day production language.

> Nothing really approach this level [of what] we can do with formal methods

not actually yet Ada can do it never mind!

Kudos to Hillel Wayne, someone who talks about Ada after actually trying it.

Useless to say I clicked the thumbs up!

[1] https://www.youtube.com/watch?=MYucYon2-lk

Ada Back in the TIOBE Top 20 (March 2025)

From: karlnyberg
Subject: Ada Back in the TIOBE Top 20 (march 2025)
Date: Wed, 2 Apr 2025 19:07:57 +0000
Newsgroups: forum.ada-lang.io

Dinosaurs indeed…

There be DRAGONS!

https://www.tiobe.com/tiobe-index/

From: zertovitch
Date: Wed, 2 Apr 2025 21:56:32 +0000

…and in the Top 15 in the PYPL index [1], possibly for the first time.

What's happening?

What is Bill Gates doing?!

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

From: pyj
Date: Sat, 5 Apr 2025 01:12:02 +0000

It does seem like the outreach has improved, but TIOBE doesn't make sense to me. I have a hard time believing C++ has a rating 3-4 times higher than JavaScript and 50 times higher than TypeScript. If you look at job boards or GitHub that seems inverted.

Languish [1] looks at the number of GitHub issues and GitHub pull requests which makes more sense, and Ada is 155 (it was 163 in 2022).

Looking at GitHub's recently updated [2] to see the number of projects:

• Zig: 5,600

• Pascal: 47,900

• Rust: 396,000

• Go: 1,100,000

• C++: 3,600,000

• Typescript: 4,200,000

• Javascript: 22,300,000

• Ada: 4,600

Don't worry about the charts and just have a good time writing cool projects.

[1] https://tjpalmer.github.io/languish/#y=mean&weights=issues%3D1%26pulls%3D1%26stars%3D0%26soQuestions%3D0&names=ada

[2] https://github.com/search?q=language%3AAda+&type=repositories&s=updated&o=desc

Ada Practice

Dimensional Analysis

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Subject: Dimensional Analysis
Date: Sat, 5 Apr 2025 07:17:50 +0000
Newsgroups: comp.lang.ada

From time to time, I come across ideas for introducing additional checking of operands in expressions that go beyond mere types, that try to take into account the actual dimensions of the units involved. The intention is that this will reduce the likelihood of expressions that, while they may be correct in conventional type-compatibility terms (all the numbers might be of compatible floating types), can be flagged if they fail a dimensional-analysis check.

I don't think it's practical to introduce new types for every single unit you want to distinguish, since that would lead to a combinatorial explosion in the necessary conversion operators. It's far more natural to build up the units in terms of basic primitives like the SI base units, and allow derived units to be expressed in terms of multiplication and division of (powers of) the base units.

Some examples, in pseudocode:

```
type velocity is units(metres / seconds);
type energy is units(kilograms * metres *
    metres * seconds / seconds);
kinetic_energy : energy;
particle_velocity : velocity;
particle_mass : units(kilograms);
```

Now suppose you have an expression like

```
kinetic_energy := 0.5 * particle_mass *
particle_velocity ** 3;
```

you can immediately see that is wrong, because the dimensions in the source and destination of the assignment don't match up. That power 3 is probably a typo, and should be a 2:

```
kinetic_energy := 0.5 * particle_mass *
particle_velocity ** 2;
```

Of course, this won't catch errors like multiplication/division by incorrect constant factors. Dimensional analysis is necessary for an expression to make physical sense, but it is not sufficient.

At first sight, this can seem useful. But you have to be careful not to carry it too far. For example, one scheme I saw wanted to specify separate units for angles so that the arguments and results of trig functions would be expressed in incompatible units. That I didn't think was a good idea.

So what do you think of this idea? Is it already in use in a significant way in any Ada code? I figured if any language would be looking at new ways of writing safer code, it would be the Ada crowd.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 5 Apr 2025 08:20:14 +0000

> So what do you think of this idea?

Static dimensionality checks are implemented by the GNAT compiler, see

https://gcc.gnu.org/onlinedocs/gnat_ugn/Performing-Dimensionality-Analysis-in-GNAT.html

For engineering purposes static checks are worthless. So in practice constrained types and dynamic checks are used instead. See

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

Atomic Increment Benchmarks

From: dmitry-kazakov
Subject: Atomic Increment Benchmarks
Date: Thu, 17 Apr 2025 09:03:03 +0000
Newsgroups: forum.ada-lang.io

I wrote a small benchmark program to compare three methods of implementing atomic increment in Ada:

• Classic Ada protected object

• System.Atomic_Operations.Integer_Arithmetic (since Ada 2022)

• GCC built-in

Here is the code:

```
pragma Ada_2022;

with Ada.Calendar; use Ada.Calendar;
with Ada.Text_IO;  use Ada.Text_IO;
with Interfaces.C; use Interfaces.C;

with System.Atomic_Operations.
Integer_Arithmetic;

procedure Lock_Free_Benchmark is

   function Add
         (  Ptr : access Integer;
            Val : Integer;
            Memorder : int := 0
         ) return Integer;
   pragma Import (Intrinsic, Add,
   "__atomic_add_fetch_4");

   protected Protected_Integer is
      procedure Increment;
      function Get return Integer;
   private
      Value : Integer := 0;
   end Protected_Integer;

   protected body Protected_Integer is
      procedure Increment is
      begin
         Value := Value + 1;
      end Increment;
      function Get return Integer is
      begin
         return Value;
      end Get;
   end Protected_Integer;

   type Atomic_Integer is new Integer with
   Atomic;
```

```
   package Atomic_Integers is
      new System.Atomic_Operations.
      Integer_Arithmetic (Atomic_Integer);
   use Atomic_Integers;

   Start : Time;
   T1, T2, T3 : Duration;
   X : aliased Atomic_Integer := 0;
   Y, Z : aliased Integer := 0;
   Times : constant := 100_000;

   procedure Report (T : Duration;
   Text : String) is
   begin
      Put (Text);
      Put (Duration'Image (T));
      Put (Integer'Image (Integer
      ((T * 1000_000_000) / Times)));
      Put ("ns ");
      New_Line;
   end Report;

begin
   Start := Clock;
   for I in 1..Times loop
      Protected_Integer.Increment;
   end loop;
   T1 := Clock - Start;

   Start := Clock;
   for I in 1..Times loop
      Atomic_Add (X, 1);
   end loop;
   T2 := Clock - Start;

   Start := Clock;
   for I in 1..Times loop
      Z := Add (Y'Access, 1);
   end loop;
   T3 := Clock - Start;

   Put_Line ("Method          Total       Time");
   Report (T1, "protected object: ");
   Report (T2, "atomic integer:   ");
   Report (T3, "gcc built-in:     ");

end Lock_Free_Benchmark;
```

As expected the second two are almost the same. Protected object is 2/3 times slower, which is quite good in my view.

Here are the measurements.

Debian:

Method	Total	Time
protected object:	0.000938000	9ns
atomic integer:	0.000536000	5ns
gcc built-in:	0.000382000	4ns

Windows:

Method	Total	Time
protected object:	0.001584700	16ns
atomic integer:	0.000534000	5ns
gcc built-in:	0.000384800	4ns

Interestingly, Linux is faster, maybe because it uses simpler locks than Windows. Remember discussion on external protected action which blocks under Linux but does not under Windows? It seems that Windows is slower for that same reason.

From: sidisyom
Date: Sun, 20 Apr 2025 21:13:22 +0000

Thanks for sharing, out of curiosity what compiler flags did you use for the executables? (assume the same in both machines?)

From: dmitry-kazakov
Date: Mon, 21 Apr 2025 08:58:06 +0000

-O2, however in this case it seems to have no visible effect.

From: Fabien.C
Date: Thu, 24 Apr 2025 14:51:27 +0000

Did you check the generated assembly code? GNAT will use atomic operations for protected objects when possible.

From: dmitry-kazakov
Date: Thu, 24 Apr 2025 15:50:23 +0000

Good point! It indeed does this when /Lock_Free/ aspect is used. If I change the line 18 to:

```
   protected Protected_Integer with
   Lock_Free is
```

Then the results look like this under Windows:

Method	Total	Time
protected object:	0.000658100	7ns
atomic integer:	0.000531200	5ns
gcc built-in:	0.000382000	4ns

Of course, in real applications a lock-free implementation would be almost never possible, especially on lousy RISC hardware. But it is great to have this option.

Generalized Private Extension ("is Record with Private")

From: LionelDraghi
Subject: Generalized Private Extension ("is Record with Private")
Date: Fri, 18 Apr 2025 22:37:32 +0000
Newsgroups: forum.ada-lang.io

There was here an interesting discussion [1] about setters and getters, and choosing between hiding or exposing data at record type level.

I sometimes have the in-between situation.

If I want to create a type Person, exposing all fields except the age, I have to either make a bunch of setters and getters, or make two types, one being private (with other annoying constraints if they are in the same package due to the “premature” use of the type).

Let’s give an example in a language that have made the (wrong) choice “Everything is a class”:

```
class Person {
  public name: string;
  // The name is publicly accessible
  private age: number;
  // The age is accessible only within the
  // Person class
constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }

  public isAdult(): boolean {
    return this.age >= 18;
  }
}
```

What would be nice in Ada to have the ability to complete a type in the private part, something like:

```
package Person is

   type Person is record with private
      Name : Unbounded_String;
   end record;
   -- "record with private" is clear enough

   -- or
   type Person is record
      Name : Unbounded_String;
   with private;
   -- More natural syntax to me, but possible
   -- confusion with predicates,
   -- and taboo "end record" dropping :-)

   function Is_Adult (P : Person) return
   boolean is (P.Age >= 18);

private
   type Person private record is
      Age : Natural;
   end record;
   -- It would be strange to have here a
   -- normal record type declaration
   -- that wouldn't tell to the reader that other
   -- fields are already declared.
   -- This is why there is an explicit "private
   -- record is",
   -- close enough to what it actually means
   -- ("private additional part of the record is").
   -- But you could object that it is already the
   -- case (not having all the fields at hand)
   -- with tagged type, and that it not worth
   -- creating a new syntax for that reason.

end Persons;
```

From the semantic point of view, it would be a normal private type, with the same constraints applying on the partial view, except that public fields are in direct access, like if Setters and Getters where provided.

And for the sake of unification, this could also apply to tagged type private extension.

Does this proposal make sense?

[1] https://forum.ada-lang.io/t/moving-a-record-to-private/1590/3

From: pyj
Date: Sat, 19 Apr 2025 00:50:45 +0000

Yes, it makes sense conceptually, but no, I wouldn’t want it and don’t ever remember a case where I’ve done something like what you’ve shown.

Forcing types to be fully exposed or private forces a distinct separation between data-only types without behavior and opaque types which maintain

invariants. Ada conceptually makes this simpler by making it such that you can tell from a single line of declaration whether a type is data or an opaque type only manipulated through subprograms.

From: Nordic_Dogsledding
Date: Sat, 19 Apr 2025 07:04:54 +0000

Is this what you're looking for? What is it that you don't like in this design and want to improve?

```
with Ada.Strings.Unbounded;
use Ada.Strings.Unbounded;

package Mixture is
  type Hidden is private;

  type Person is record
    Name: Unbounded_String;
    More: Hidden;
  end record;

  function is_Adult (P: Person) return
  Boolean;

private

  type Hidden is record
    Age: Natural;
  end record;

  function is_Adult (P: Person) return
  Boolean is (P.More.Age >= 18);

end Mixture;
```

From: LionelDraghi
Date: Sat, 19 Apr 2025 16:27:15 +0000

> Forcing types to be fully exposed or private forces a distinct separation between data-only types without behavior and opaque types which maintain invariants. Ada conceptually makes this simpler by making it such that you can tell from a single line of declaration whether a type is data or an opaque type only manipulated through subprograms.

OK, I agree.

> Is this what you're looking for?

This is clear, short, almost perfect

I have oversimplified my case, and today I can't remember how this proposal could have helped with my problem!

I have a self referential structure, fully exposed in a spec (bbt/src/bbt-documents.ads at [1]).

Jeffrey [Carter] wrote (Papers/Self_Ref_No_Access.pdf at [2]) and talked about that problem.

The solution seems to be:

- Status Quo, I let the details exposed, no private part;
- Using Vector of access type instead of a Vector of the type works: I wrote the code, it works, I was able to split this big package in a child hierarchy by using limited and private with.

But I'm not a big fan because of the access type;

- Using an Ada.Containers.Indefinite_Holders instance to hold the Vectors is more complex than needed;
- Using tagged record instead of record for the nodes is OK (and actually make sense in my case, because all nodes share some common properties), but, first, having Vectors of Node'Class instead of Vector of Node make the code kind of lying on my design intents, and I have the impression of inviting possible errors…

None of the solutions is really nice, but I'm now experimenting with the last one.

Anyway, the night has gone, and I can't remember how this proposal could have helped with my problem.

[1] https://github.com/LionelDraghi/bbt/blob/1061e62626caeeb5511e7643360c580927083ffb/src/bbt-documents.ads

[2] https://github.com/jrcarter/Papers/blob/b579a6550bdbaa30bd3b7fe008f6c485dfa9f9dc/Self_Ref_No_Access.pdf

Generics and Compilation Time

From: Shuihuzhuan
Subject: Generics and Compilation Time
Date: Sun, 1 Jun 2025 10:06:38 +0000
Newsgroups: forum.ada-lang.io

I have a really simple generic package like that (spec only here):

```
pragma Ada_2022;

with Ada.Containers.Indefinite_Vectors;
with Ada.Strings.Text_Buffers;

generic
   type Element () is tagged private;
   with procedure Element_Image (Output:
   in out Ada.Strings.Text_Buffers.
   Root_Buffer_Type'Class;
   Item : Element) is <>;
package Gir_Reader.Generic_Lists is

   use type Element;

   package Element_Lists is new
   Ada.Containers.Indefinite_Vectors
     (Index_Type   => Positive,
      Element_Type => Element);

   type Element_List is new
   Element_Lists.Vector with null record
     with Put_Image => Image;

   procedure Image
     (Output : in out Ada.Strings.Text_Buffers.
        Root_Buffer_Type'Class;
      Item : Element_List);

end Gir_Reader.Generic_Lists;
```

Without using it, compilation of my whole program takes around 11s. Each time I instantiate this package, I get 3s more compilation time. Now, I'm around 25s with 5 instantiations.

If instead of using a generic I just repeat the code, the compilation time penalty is negligible.

Are there reasons for such compilation penalty using generics?

From: Irvise
Date: Sun, 1 Jun 2025 11:56:06 +0000

Generics are known to not be that efficient with GNAT. However, the time penalty that you are getting is way too large to be acceptable.

Could you provide information about what compiler version you are using and which flags are enabled?

From: Shuihuzhuan
Date: Sun, 1 Jun 2025 13:39:41 +0000

Thanks Irvise.

Without my own generic package, the time penalty is not as negligible as I thought. I had just tried with 2-3 copy-pastes. With the whole use cases in my code, I get around the same compile time penalty.

So this compilation time is from new Ada.Containers.Indefinite_Vectors.

My computer is quite old (AMD Phenom II X3 from 2009). That may explain part of the problem. But well, 3s for that :->

I'm on archlinux using Alire with standard flags.

```
Ada_Compiler_Switches :=
External_As_List ("ADAFLAGS", " ");
Ada_Compiler_Switches :=
Ada_Compiler_Switches &
 (
  "-Og" -- Optimize for debug
 ,"-ffunction-sections"
  -- Separate ELF section for each function
 ,"-fdata-sections"
  -- Separate ELF section for each variable
 ,"-g" -- Generate debug info
 ,"-gnatwa" -- Enable all warnings
 ,"-gnatw.X" -- Disable warnings for
                --No_Exception_Propagation
 ,"-gnatVa" -- All validity checks
 ,"-gnaty3" -- Specify indentation level 3
 ,"-gnatya" -- Check attribute casing
 ,"-gnatyA" -- Use of array index numbers
               -- in array attributes
 ,"-gnatyB" -- Check Boolean operators
 ,"-gnatyb" -- Blanks not allowed at
              -- statement end
 ,"-gnatyc" -- Check comments
 ,"-gnaty-d" -- Disable check no DOS line
               -- terminators present
 ,"-gnatye" -- Check end/exit labels
 ,"-gnatyf" -- No form feeds or vertical
              -- tabs
 ,"-gnatyh" -- No horizontal tabs
 ,"-gnatyi" -- Check if-then layout
 ,"-gnatyI" -- check mode IN keywords
 ,"-gnatyk" -- Check keyword casing
 ,"-gnatyl" -- Check layout
 ,"-gnatym" -- Check maximum line length
 ,"-gnatyn" -- Check casing of entities in
              -- Standard
 ,"-gnatyO" -- Check that overriding
```

```
                -- subprograms are Ç
                -- explicitly marked as such
        ,"-gnatyp" -- Check pragma casing
        ,"-gnatyr" -- Check identifier
                -- references casing
        ,"-gnatyS" -- Check no statements
                -- after THEN/ELSE
        ,"-gnatyt" -- Check token spacing
        ,"-gnatyu" -- Check unnecessary blank
                -- lines
        ,"-gnatyx" -- Check extra parentheses
        ,"-gnatW8" -- UTF-8 encoding for wide
                -- characters
    );
```

and

```
toolchain.use.gnat=gnat_native=14.2.1
toolchain.use.gprbuild=gprbuild=22.0.1
toolchain.external.gnat=false
```

Have a nice day!

From: jere
Date: Sun, 1 Jun 2025 15:28:54 +0000

Out of curiosity, if you remove the Put_Image function and the aspect for it on your main type does that change the timing at all? Asking because Put_Image is a relatively new feature, so maybe the implementation for it isn't quite optimized.

Also are you using HDD or SSD type drives? The binding stage is going to generate files and if your antivirus scans them (a lot do by default unless disabled), that might be getting slow on an HDD type drive. Not saying it's an acceptable time loss, but trying to isolate the cause.

From: Shuihuzhuan
Date: Sun, 1 Jun 2025 16:11:12 +0000

Thanks for the advice.

Without the Put_Image aspect, I get almost the same compilation time (37s vs 39s with 10 instantiations).

I'm using an SSD drive and no antivirus (I'm on linux).

From: JC001
Date: Tue, 3 Jun 2025 09:31:55 +0000

> Without my own generic package, the time penalty is not as negligible as I thought. I had just tried with 2-3 copy-pastes. With the whole use cases in my code, I get around the same compile time penalty.

So this has nothing to do with generics; it's due to the amount of code that you're compiling, which is roughly the same whether you use generics or code duplication, since GNAT does instantiation by macro expansion. What might save you time is using a compiler that implements shared-code generics (DEC Ada or Janus Ada).

> it's nicer to have 10s compilation time instead of around 40s

The first Ada compiler I used was the Rolm/Data General compiler, which took about 10 minutes to compile a "Hello, World!" program, so I don't know what you're complaining about.

From: DirkCraeynest
Date: Tue, 3 Jun 2025 09:51:34 +0000

> The first Ada compiler I used was the Rolm/Data General compiler, which took about 10 minutes to compile a "Hello, World!" program, so I don't know what you're complaining about.

Mind you, for the younger people here, what Jeffrey is talking about is a compiler dating from the early 1980's (1983 or 1984 IIRC). In fact, that compiler got Validation Certificate nr. 2, after NYU's Ada/Ed that got the first certificate (but the latter was an interpreter written in Setl, a language based on sets and interpreted as well, i.e. two levels of interpretation). So ROLM/Data General was the first validated Ada 1983 /compiler/. Note that it was still in serious use at least until 2006. Been there, done that…

Historic info added to avoid Jeffrey's posting would be used to falsely claim "Ada is slow" [sic]. 40+ years ago, compiling and running a Java or Rust program would have taken infinitely longer…

From: Shuihuzhuan
Date: Tue, 3 Jun 2025 11:38:19 +0000

> So this has nothing to do with generics; it's due to the amount of code that you're compiling, which is roughly the same whether you use generics or code duplication, since GNAT does instantiation by macro expansion.

It somehow has to do. It's just not my own generic package but the indefinite_vector generic that takes time to compile!

From: JC001
Date: Wed, 4 Jun 2025 09:28:41 +0000

> It's just not my own generic package but the indefinite_vector generic that takes time to compile!

Well, of course. An instance of Indefinite_Vectors is a pretty large package. But if you hand-instantiate it, it will still take about as long to compile.

From: dmitry-kazakov
Date: Wed, 4 Jun 2025 09:46:41 +0000

From my experience the package length had little to do with the problem. I had a lot of small generics instantiated for various scalar types (signed, unsigned x size). It blew the GNAT Pro out.

I would propose to consider instead of generics in generics to look if

- child generics
- formal generic instance as an argument

do better.

(Even better, avoid generics if you can, and use tagged types instead!)

From: dmitry-kazakov
Date: Tue, 3 Jun 2025 13:02:05 +0000

> So this has nothing to do with generics; it's due to the amount of code that you're compiling, which is roughly the same whether you use generics or code duplication

It has all to do with generics. GNAT implementation is extremely slow and consumes incomprehensible amounts of memory. It has become much better in recent decades. The production code that actively used generics compiled several /days/ then and I could not compile it on a 32-bit system. It ran out of memory!

I think AdaCore added a lot of optimisation since then, which does not kick in this concrete case because of new Ada features.

From: cantanima
Date: Sun, 1 Jun 2025 23:02:32 +0000

You mention the compile time, but how does the /execution/ time change, if at all?

If the execution time is faster, then the longer compile time might be an acceptable tradeoff.

From: Shuihuzhuan
Date: Mon, 2 Jun 2025 05:12:44 +0000

It doesn't change anything at runtime. I was just surprised to have a 3s compilation time added each time I added an instantiation. And when developing, it's nicer to have 10s compilation time instead of around 40s :->

From: jcmoyer
Date: Tue, 3 Jun 2025 02:56:35 +0000

I have found that it helps immensely to pre-instantiate expensive-to-compile generics outside of the file where you use them. This prevents them from being re-instantiated every time you compile modified code.

For example, a string_vectors.ads might only contain:

```
with Ada.Containers.Indefinite_Vectors;

package String_Vectors is new
Ada.Containers.Indefinite_Vectors
  (Index_Type => Positive,
   Element_Type => String);
```

Then String_Vectors will be instantiated once, cached, and reused across compilations (at least until you modify the generic body).

Specifying Major Order for Packed Matrix

From: mosteo
Subject: Specifying Major Order for Packed Matrix
Date: Sun, 8 Jun 2025 19:01:32 +0000
Newsgroups: forum.ada-lang.io

See this declaration for 8 bits arranged in a matrix of 4x2, indexed by (row, column):

```
type Cell_Matrix is array (1..4, 1..2) of
Boolean with
   Component_Size => 1,
   Size => 8,
   Convention => Fortran;
```

I was hoping that this would give column-major order but Unchecked_Conversion reveals row-major order.

After a few searches and ARM digging I could not find an answer to the following:

- Can a matrix major order be specified somehow? [Couldn't find a way]
- If not, are Ada matrices guaranteed to be row-major? [I think not]

I have several workarounds at hand, no need to suggest alternatives. I'm interested on the normative behind it only. Thanks!

From: dmitry-kazakov
Date: Sun, 8 Jun 2025 20:11:09 +0000

Hmm, considering this:

```
with Ada.Text_IO; use Ada.Text_IO;
with Interfaces.Fortran; use
Interfaces.Fortran;

procedure Test is
   type Cell_Matrix is array (1..4, 1..2) of
   Real with Convention => Fortran;

   type Matrix is array (1..2, 1..4) of Real;

   A : Cell_Matrix := ((1.1, 1.2), (2.1, 2.2),
   (3.1, 3.2), (4.1, 4.2));
   B : Matrix;
   pragma Import (Ada, B);
   for B'Address use A'Address;
begin
   for I in A'Range (1) loop
      for J in A'Range (2) loop
         Put (Real'Image (A (I, J)));
         Put (' ');
      end loop;

      New_Line;
   end loop;
   New_Line;
   for I in B'Range (1) loop
      for J in B'Range (2) loop
         Put (Real'Image (B (I, J)));
         Put (' ');
      end loop;
      New_Line;
   end loop;
end Test;
```

The output is:

```
 1.10000E+00  1.20000E+00
 2.10000E+00  2.20000E+00
 3.10000E+00  3.20000E+00
 4.10000E+00  4.20000E+00

 1.10000E+00  2.10000E+00  3.10000E+00
 4.10000E+00
 1.20000E+00  2.20000E+00  3.20000E+00
 4.20000E+00
```

It looks correct to me.

From: JC001
Date: Mon, 9 Jun 2025 11:00:16 +0000

It is only Implementation Advice that arrays with Convention Fortran use column-major order, so a compiler does not have to do so. However, I know that GNAT does so, at least for unpacked arrays of numbers. You're packing your array into a single byte, and that may conflict with or take precedence over the convention. I tried

```
with Ada.Text_IO;
with Ada.Unchecked_Conversion;

procedure Fortran_Test is
   type Cell_Base is array (1 .. 4, 1 .. 2) of
   Boolean;
   type Cell_Fortran is new Cell_Base with
   Convention => Fortran;
   type Cell_Packed  is new Cell_Base with
   Component_Size => 1, Size => 8;
   type Cell_PF is new Cell_Packed with
   Convention => Fortran;

   B : constant Cell_Base := ( (False, False),
   (False, True), (True, False), (True, True));
   F : constant Cell_Fortran := Cell_Fortran
   (B);
   PB : constant Cell_Packed  :=
   Cell_Packed (B);
   PF : constant Cell_PF      := Cell_PF (F);

   type Byte is mod 256;

   function To_Byte is new
   Ada.Unchecked_Conversion (Source =>
  Cell_Packed, Target => Byte);
   function To_Byte is new
   Ada.Unchecked_Conversion (Source =>
   Cell_PF, Target => Byte);
begin -- Fortran_Test
   Ada.Text_IO.Put_Line (Item => "PB =" &
   To_Byte (PB)'Image & " PF =" & To_Byte
   (PF)'Image);
end Fortran_Test;
```

with GNAT and got 216 for both. With ObjectAda I get PB = 216 PF = 2; the value for PF is clearly wrong.

From: mosteo
Date: Mon, 9 Jun 2025 12:14:43 +0000

So it's the packing that in the case of GNAT is nullifying the Fortran convention.

Anyway, as long as those are recommendations and there's no way to be sure that the compiler either gives you what you want or a compilation error, I'm not happy with relying on any behavior. And GNAT at least is not warning about not honoring the convention.

Thank you both for your tests.

From: jere
Date: Mon, 9 Jun 2025 14:28:37 +0000

It might be worth putting in a report with AdaCore and see if it is intended. Even though it is implementation advice, I always took that to be more of the advice was "to either provide the convention or not and if you do here are the rules". It would be odd to have the convention only if it doesn't actually map to the language types correctly in some situations (without warnings).

From: Irvise
Date: Mon, 9 Jun 2025 17:46:52 +0000

This goes beyond Ada.

Fortran does have booleans and arrays of booleans. HOWEVER, it *does not have bit arrays (packed arrays of booleans)*. Bit arrays in Fortran, as expected by the standard and by its users, are Arrays of integers on which bit operators can be performed (or just a single INT*8/16/…).

Therefore, what you are trying to do here is illegal in Fortran and its behaviour is either undefined or implementation defined. I think this is good. Though true, GNAT could issue a warning!

Also, I would like to see more Fortran/Ada interop

From: OneWingedShark
Date: Mon, 9 Jun 2025 18:33:16 +0000

This is not a very satisfactory answer: the notion of an array of integers —even constrained to the range 0 to 1, inclusive— indexed in row- or column-major ordering is separate and distinct from the mere implementation-detail of how much space said integers take up.

From: Irvise
Date: Mon, 9 Jun 2025 19:00:04 +0000

Yes, I agree, but the problem remains. I also did not want to focus too much on the space required for the integers… Though I understand that the underlying memory should be the same, so Ada/GNAT could try a bit harder or just be honest and say that this is a mistake

From: mosteo
Date: Tue, 10 Jun 2025 11:37:59 +0000

> HOWEVER, it *does not have bit arrays (packed arrays of booleans)*

This is a valid point, but I'm really not interested in interoperating with Fortran, it was just the only thing I found that could affect the component layout. I would be perfectly happy with something like

```
type Matrix is array (1 .. 4, 1 .. 2) of Boolean
with Storage_Layout => Column_Major;
```

I find it interesting that you can be precise (and get guarantees) with record representation clauses, but there seems to be no similar capabilities for arrays.

From: dmitry-kazakov
Date: Tue, 10 Jun 2025 12:54:16 +0000

The next step would be supporting packed symmetrical matrixes and three-diagonal matrices, video frame buffers etc…

From: JC001
Date: Wed, 11 Jun 2025 09:52:54 +0000

Note that even for something like

```
type Bit_Map is array (1 .. 8) of Boolean
with Component_Size => 1, Size => 8;
```

the language doesn't specify which index corresponds to which bit position. If you need to use indices to access specific bit positions, it's probably best to use a private type with appropriate operations.

Ada Frontiers

Plan for Next Version of International Ada Standard

From: sttaft
Subject: Plan for Next Version of International Ada Standard
Date: Fri, 18 Apr 2025 15:00:53 +0000
Newsgroups: forum.ada-lang.io

As the ISO WG9 Ada Rapporteur Group (ARG) finishes up its work on Ada 2022 and its first "corrigendum", we are beginning the next major revision cycle. At this point we are looking for brainstorming and prioritizing of the next set of features for Ada. If you want to provide some inputs, please visit

https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues/134

From: LionelDraghi
Date: Mon, 21 Apr 2025 23:50:35 +0000

As it happens, Brian Goetz, the lead architect of Java just gave a conference titled " Where is the Java language going?"

Very interesting to see the process, the major identified topics, the sources of inspiration (which do not include Ada, and yet!), to see how initial choices weigh in 30 years later, to hear the discourse on 'we went too far' with objects, the consequences of hardware evolution that now makes the widespread use of pointers costly, etc.

By the way, on this last point, it seems to me that with its more favorable stack vs. heap balance and its ability to use fewer pointers than other languages, Ada should position itself in the data-oriented wave.

It would be quite amusing if, decades after being considered heavy and slow, Ada made a name for itself in the arena of high-performance languages!

From: pmnw
Date: Tue, 22 Apr 2025 19:59:24 +0000

Thanks, I will watch that with great interest.

You make a very good point with regards to the increasing focus on data flows and computing rather than just applications.

It's interesting to consider Ada as a language that enables that sort of programming.

Arguments about definitions of what is "data-orientation" aside, I have my doubts given the shape of Ada's standard library which offers very little data-oriented functionality - sorts, finds, min, max, etc. are far from being sufficient for work with "big" data.

To make the thing worthwhile, it would have to be competitive in offering high-level abstractions lifted from functional programming languages because serious data processing cannot be done without them (~ the language expressing the data flow through functional transforms and saving users from writing raw loops).

For example, C++ has lately went full on ranges and views with operations such as applies, folds, filters, joins, concats, splits, zips, chunks, slides, strides, and so on, along with piping that together make it feel almost like programming in R's Tidyverse.

It has generators and coroutines and a state-of-the-art random library.

It all works well with Stepanov's original algorithm designs like rotations, etc.

It could absolutely be done in Ada and I expect the result would be much more elegant and beautiful, I have no doubt, but I cannot imagine Ada getting a fraction of this functionality.

From: OneWingedShark
Date: Wed, 23 Apr 2025 03:15:06 +0000

> Arguments about definitions of what is "data-orientation" aside, I have my doubts given the shape of Ada's standard library which offers very little data-oriented functionality - sorts, finds, min, max, etc. are far from being sufficient for work with "big" data.

There is some argument that "big data" is more often than not "stupid data" — like using textual '1' and '0' to represent bits, for every bit in a byte, for every byte in a word. (Not entirely wrong: IIUC, the recent killer jump in AI came from dropping float-sizes; I expect it would have been even cheaper if they went to fixed-point.)

"The Industry" idiocy aside, you are right that there ought to be some development of a suite of algorithms and, I presume, data-structures… Welcome to 1985. — It was always the idea that an /Ada Programming Support Environment/ [APSE] would have such functionality; see:

- Adabase: a data base for Ada programs [1] (1983)
- Workspaces and experimental databases: Automated support for software maintenance and evolution [2] (1987)
- The importance of Ada programming support environments [3] (1982)
- A Revised STONEMAN for Distributed Ada Support [4] (1983)

But there's a huge point of wisdom here that you may be missing: these capabilities were wisely kept from the /Ada Programming Language/ Standard. — This isn't to say that we shouldn't have good, well-built libraries, but that there shouldn't be a standard library that includes everything-and-the-kitchen-sink. (Which is perhaps the /other/ "we went too far" of Java.)

[1] https://dl.acm.org/doi/pdf/10.1145/3304133.3304141

[2] https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f5980272e4b21a5e7c44e0c1b1af3be9beae5e38

[3] https://dl.acm.org/doi/pdf/10.1145/1500774.1500815

[4] https://apps.dtic.mil/sti/tr/pdf/ADA137940.pdf

From: pmnw
Date: Wed, 23 Apr 2025 09:21:53 +0000

> But there's a huge point of wisdom here that you may be missing: these capabilities were wisely kept from the /Ada Programming Language/ Standard. — This isn't to say that we shouldn't have good, well-built libraries, but that there shouldn't be a standard library that includes everything-and-the-kitchen-sink.

I agree, I made this exact point… implicitly, and really there is just too much that can be put behind "data-driven" or "data-oriented" for a meaningful discussion without numerous preconditions.

If Ada users want more performance à la @LionelDraghi just above, then I'm all for it, but it won't enable what I understand as "data-oriented" programming.

There is also a chasm between offering performance and people making use of it.

More wrt the /Plan for next version of International Ada Standard/

Looking at the JavaOne video - the language is getting functional imports like algebraic data types and pattern matching and syntax sugars. Working with data requires an ergonomic DSL and they seem to think these features will serve as such. It may work for them, but Ada seems to me in need of shaving and polishing more than inflation.

So, polishing what's there - yes, opening new fronts - no.

I'm saying this as an FP appreciator. Until recently I thought it would be nice for Ada to also import some of the functional features, but after some more thought, I'm thinking that wouldn't work. It would further complicate an already complicated language.

Ideas I like:

- unicode
- cleanup & annex J
- libraries
- searchable documentation

Unicode and language cleanup should be solved on the language-stdlib level.

In a perfect world, simplifying the language by removing obsolescent or duplicated features would be the utmost priority. “Stop adding stuff, start removing it.”

Ideas I dislike (Github):

- allowing instantiation at the point of use, and with more parameters inferred - against Ada design.
- safe use of pointers, potentially including support for pointer ownership - I’d rather be reading about banning pointers and null, and in any case, you won’t out-Rust Rust, so why bother.
- standard packages for XML and JSON.

The last point really makes you think… I’m unaware of any algorithms libraries in Ada, but there’s a proposal to include XML in the standard library. The standard library offers no tools for high-level data structure transforms beyond the most pedestrian operations like find, and it’s not being discussed - but it could soon have XML.

That’s interesting and seems to confirm the emergent consensus (what @Lucretia said in the GitHub discussion) that Ada is a language for embedded development.

From: Lucretia
Date: Wed, 23 Apr 2025 10:17:56 +0000

> safe use of pointers, potentially including support for pointer ownership - I’d rather be reading about banning pointers and null, and in any case, you won’t out-Rust Rust, so why bother

Adding a modern profile where other pointer facilities are the default, like nullable (easier to add?) or optional (requires fp afaik) instead of access types.

From: jere
Date: Wed, 23 Apr 2025 17:01:51 +0000

> safe use of pointers, potentially including support for pointer ownership - I’d rather be reading about banning pointers and null, and in any case, you won’t out-Rust Rust, so why bother

I think if you eliminated access types entirely, then with the current Ada it would be shooting yourself in the foot. There’s very little language support in managing collections of limited objects. Right now you have to rely on access types or global variable declarations if you want to do complex stuff with limited types. Also there’s definitely a middle ground between Rust level semantics and where we are now. I think anything that removes the need for access types or at least makes them safer to use in common situations is a good thing to consider.

I do think that the standard library and the language should increase the ability to avoid pointers in more situations (it already covers a lot of area in this aspect, but still there are outliers, and some of the existing solutions are not great / readable / maintainable).

> In a perfect world, simplifying the language by removing obsolescent or duplicated features would be the utmost priority. “Stop adding stuff, start removing it.”

I agree here about removing things (I don’t think we stop adding stuff if we need it), but I will say based on my experience reading the ARG discussions, it seems very unlikely. A lot of folks on the committee are worried about breaking older versions of the language, which for me makes little sense, since people using older versions of the language can use the compilers they always used without the new features, or restrict themselves to an older version of the language via compiler switches/pragmas. I feel like compiler vendors can make the business decision based on their users if they want to maintain / update those older versions or just move forward (or do both).

From: sttaft
Date: Wed, 23 Apr 2025 17:48:26 +0000

> I agree here about removing things [...]

We do move things into Annex J, the Obsolescent Features annex, when we believe we have replaced them with something clearly better. This can make the description of the language simpler (presuming you ignore Annex J!), without necessarily breaking old code.

At some point if a feature has been in the Obsolescent Features Annex for several language versions, I suspect some compiler implementors might stop supporting it, though if they have customers using old versions, they might more likely just presume that customers who care can use the Restriction No_Obsolescent_Features (see RM 13.12.1(4/3)).

From: pmnw
Date: Wed, 23 Apr 2025 17:57:04 +0000

> I think if you eliminated access types entirely, then with the current Ada it would be shooting yourself in the foot.

Of course, I am just being grumpy about the sudden need for smart pointers or borrowing in a 40-yo language. Perhaps this is politics, idk.

I’m mostly worried about the language playing catch-up with Rust at the cost of more and more additions. Even then, if more is better, I’d prefer additions to be higher-level than memory management facilities.

From: Lucretia
Date: Tue, 22 Apr 2025 20:29:03 +0000

> on ‘we went too far’ with objects

They definitely did, Java really is spaghetti code where you never know what object you’re using, it’s a mess really and I really think caused the modern hatred of oop.

> Ada made a name for itself in the arena of high-performance languages!

Since when was Ada /not/ high-performance??

From: LionelDraghi
Date: Tue, 22 Apr 2025 20:36:14 +0000

> Since when was Ada /not/ high-performance??

I was speaking of perception only!

From: LionelDraghi
Date: Tue, 22 Apr 2025 22:01:20 +0000

> Arguments about definitions of what is “data-orientation” aside

I was thinking of the performance aspect of Data Oriented Design.

For example, the AdaCore proposal of a Max_Size aspect for classwide type [1]

The Motivation chapter says “Being able to use classwide types where definite types are required in Ada, which in turns allows to either use tagged types where they couldn’t be used before”, and then discuss the benefit of removing dynamic allocation, and getting tagged types more used in embedded context.

From my point of view, the main benefit would be to enable arrays of classwide instances of such types, without pointers, that is with a dramatic effect on locality and a potential huge gain in performance.

With only the “with Maxsize => x” aspect added to the declaration of the root tagged type (and maybe some other aspect to lock the class hierarchy).

I dream I could just declare

```
Team : array (1 .. 10) of Player'Class;
```

No access type, no indefinite containers: not only the performances are much better, but the code is simpler than ever.

[1] https://hackmd.io/q0NXV7J8RdiambtId8CMsg

From: jere
Date: Tue, 22 Apr 2025 22:53:22 +0000

Space performance would potentially be worse, but I can see a better speed increase. On the GNAT specific front they are adding a settable aspect Class'Size => in GCC 15 that might facilitate the above. The normal problem is that the size of Player’Class is unknown at compile time (in the general case). That new aspect allows one to set an upper bound, which I would imagine makes array type declarations possible.

From: sidisyom
Date: Sun, 27 Apr 2025 20:45:42 +0000

> I dream I could just declare

It looks like this is included [1] (see feature #2) in the latest version of GNAT Pro (with -gnatX0).

+1 for the class hierarchy lock-down. In the jvm space Kotlin has this nice feature called /Sealed Classes/ which makes all subclasses statically known during compilation time and I assume allows a compiler to apply some nice performance optimisations since it is no longer restricted by the dynamic class loading requirement of the jvm.

At the code level, it offers some nice readability benefits when casing (or in Kotlin parlance whening) on the type of a function argument.

In the jvm of course, /all/ allocations are heap-based so it is a different world there (though, that is not strictly accurate with recent (JIT) compilers).

[1] https://blog.adacore.com/lets-play-7-differences-in-ada

From: Micronian2
Date: Wed, 23 Apr 2025 18:30:37 +0000

I too think more effort should be placed on improving the existing parts of the language and less on major language additions that would take longer to get implemented (if ever, which depends on what AdaCore does). More standard libraries would be good too. I also don't want more proliferation of anonymous access types (yuck!). I absolutely do *not* want GNAT's new "class" type extension (i.e. the one that is similar to C++ classes) to become standard either.

From: sttaft
Date: Wed, 23 Apr 2025 19:17:12 +0000

> I absolutely do *not* want GNAT's new "class" type extension (i.e. the one that is similar to C++ classes) to become standard either.

I think many members of the ARG agree with you.

From: OneWingedShark
Date: Wed, 23 Apr 2025 20:42:33 +0000

> I think many members of the ARG agree with you.

I certainly don't want it: it adds no significant value for additional complexity.

The bizarre thing, though is the /*Why!?*/ — If you wanted to do something much more usable with the effort, implement an updated Lamb-Nestor IDL (which DIANA was an instance of), bringing its capabilities (and possibly syntax) more in-line with Ada/SPARK, with an eye toward producing Ada/SPARK & ASN.1, then you have something /far/ more usable, and which could be used both to generate your classes, with SPARK-annotations, and to enable robust heterogenous-system comms/interchange.

From: waleedmebane
Date: Sun, 27 Apr 2025 21:18:55 +0000

About removing things and whether Ada (or Java) is too big: It seems to me that one of the appeals of the language is that decades old software still works, and one can presumably expect code one writes today to work for decades with little change. The same seems true for Java, but it likely wouldn't be if programmers had to rely on third-party libraries to provide common features since such third-parties are usually less concerned about supporting old features than compiler writers. It's not just a matter of using an old version of the compiler necessarily since sometimes the machines and OSes that they ran on cease to be produced. (For Java, that might be okay as long as the byte-code syntax and semantics don't change.)

What if POSIX hadn't been so successful? Then, much less C code would be working today on common machines and OSes unless there were some similarly long-lived standard and library implementations for those machines (and maybe also an easy portability path).

So, it seems like the large scope of the language standard is a big part of what makes such longevity for programs possible without much maintenance burden.

SPARK/Ada

Importing C with SPARK

From: Lucretia
Subject: Importing C with Spark
Date: Thu, 24 Apr 2025 09:45:00 +0000
Newsgroups: forum.ada-lang.io

Can it be done? The docs are 4 lines and says:

> This section describes features for mixed-language programming in SPARK, covering facilities offered by Ada's Annex B.

> Package Interfaces can be used in SPARK, including its intrinsic "Shift" and "Rotate" functions.

> Other packages are not directly supported.

From: jere
Date: Thu, 24 Apr 2025 17:22:42 +0000

I don't know SPARK, but the reference manual seems to indicate both Convention and Import are supported, so my guess is that wording implies that what languages are supported depends on the vendor.

[1] https://docs.adacore.com/spark2014-docs/html/lrm/language-defined-aspects-and-attributes.html

From: Lucretia
Date: Thu, 24 Apr 2025 17:26:22 +0000

I did a quick test and it seems you can.

From: sttaft
Date: Thu, 24 Apr 2025 22:19:45 +0000

This has been changing recently, I believe, as more customers are using SPARK and C together. And as you discovered, it does work, even if it isn't "directly supported".

SPARK-verified Postfix Expression Calculator

From: pyj
Subject: Spark Verified Postfix Expression Calculator
Date: Thu, 12 Jun 2025 02:25:49 +0000
Newsgroups: forum.ada-lang.io

I got inspired by @streaksu's Ironclad work and @OneWingedShark's idea to write a verified seedForth interpreter in SPARK. That seemed a bit large as a starting SPARK project, so I wrote a SPARK verified postfix expression calculator [1].

I had a great time doing this, so I might try to fork that calculator into a simple Forth interpreter. I don't have formal methods training, but I wrote up my experience if anyone is curious [2].

I would really appreciate any guidance on the current code or related post from those with formal methods experience. The only reference I have right now is McCormick/Chapin's "Building High Integrity Applications with SPARK" book, but that has been very helpful so far.

[1] https://github.com/pyjarrett/postfix_calc

[2] https://pyjarrett.github.io/2025/06/10/postfix-calculator.html

From: Fabien.C
Date: Thu, 12 Jun 2025 07:47:36 +0000

Nice project! I'd love to see you continue

From: gast
Date: Thu, 12 Jun 2025 13:35:20 +0000

Thank you! Could you please also add the prover output to the README in [1]? This way, other non-Ada users can see what it looks like without having to install and run anything.

[1] https://github.com/pyjarrett/postfix_calc

From: OneWingedShark
Date: Thu, 12 Jun 2025 15:47:52 +0000

Very nice!

Glad I was somewhat inspiring.

(Reading the article now.)

From: mgrojo
Date: Sat, 14 Jun 2025 16:46:55 +0000

Regarding " Floating point simplifications", what I've found useful in CoAP-SPARK [1] for proving floating point properties is the COLIBRI prover. It doesn't come with the Alire distribution of gnatprove, but can be separately obtained from the GNAT Community 2021 edition. Then, just putting the binary on the path is enough to make gnatprove

find it and use it when requested by the arguments. This workflow [2] automating the proof process of my project is doing the download and extraction from GNAT Community.

[1] https://github.com/mgrojo/coap_spark

[2] https://github.com/mgrojo/coap_spark/blob/3dc855f1cf5c730b512c9c782c82973a10f95c81/.github/workflows/prove.yml

From: Irvise
Date: Sun, 15 Jun 2025 06:50:25 +0000

You can download the Colibri and Colibri2 (still highly in beta) from their website: Colibri [1]

Their repositories (where the binaries come from) are the following:

- Colibri: pub / COLIBRI · GitLab [2]
- Colibri2 (colibrics): pub / colibrics · GitLab [3]

I am pointing this out as they are open source projects that have seen quite a bit of development since the GNAT Community pack was discontinued. I believe it may also be much easier to download the binaries (or compressed binary) directly and just put it into the Path. I have tested Colibri and it did not do much better than the others, but as @mgrojo points out, each solver has their strengths, and Colibri's lies in numeric stuff rather than program flow or similar topics.

I hope this helps you all!

[1] https://colibri.frama-c.com/download.html

[2] https://git.frama-c.com/pub/colibri

[3] https://git.frama-c.com/pub/colibrics/

From: mgrojo
Date: Sun, 15 Jun 2025 14:42:33 +0000

Using the latest Colibri or Colibri2 releases from the repositories, as you mention, was my first approach, but they aren't really compatible with gnatprove yet, as AdaCore confirmed to me through the GNAT Academic program. But the tool fails silently and you don't see any difference. The working version is Colibri 2020.9 [1], but building from source is not trivial. I finally discovered that the correct version was present in the latest Community Edition. The license [2] is LGPL since 2020 [3], so there are no possible worries about using or distributing it.

[1] https://git.frama-c.com/pub/colibri//commit/ba7a7755ab9c58043e693fcadb6f3aaf16f63ad5

[2] https://git.frama-c.com/pub/colibri//blob/ba7a7755ab9c58043e693fcadb6f3aaf16f63ad5/LICENCE

[3] https://git.frama-c.com/pub/colibri//commit/03b6e2900abb24ef072541733517bb04a67260a9

From: Irvise
Date: Sun, 15 Jun 2025 19:37:10 +0000

> but they aren't really compatible with gnatprove yet, as AdaCore confimed me through the GNAT Academic program. But the tool fails silently and you don't see any difference.

Oh, that explains a few things I saw during my testing…

That is actually a bit surprising to me as all the tools are using Why3 and there are already four provers supported (Z3, alt-ergo, cvc5 and Coq with some tweaks…). Oh well… That is quite sad. I know however that AdaCore was looking/working in Colibri, which is fully confirmed by the fact that it was distributed with the Community Edition…

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

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.

July 07-10 — 4th **Summer School on Security Testing and Verification** (ST&V'2025), Brussels, Belgium. Topics include: security testing, software verification, static program analysis, dynamic program analysis, fuzz testing, etc.

July 07-11 — 52nd EATCS **International Colloquium on Automata, Languages and Programming** (ICALP'2025). Aarhus, Denmark. Topics include: all aspects of theoretical computer science, such as data structures, distributed computing, parallel computing, models of concurrent and distributed systems, models of reactive systems, principles and semantics of programming languages, program analysis, verification, and synthesis, type systems, etc.

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.

July 14-16 — 19th **International Symposium on Theoretical Aspects of Software Engineering** (TASE'2025). Limassol, Cyprus. Topics include: software architectures and design; software product lines; model-driven software engineering; software testing and quality assurance; software safety, security and reliability; reverse engineering and software maintenance; component-based software engineering; formal methods; deductive verification; model checking; theorem proving; specification languages; run-time verification and monitoring; integration of formal methods; formal methods for AI systems, and vice versa; programming language design and technology; abstract interpretation and program analysis; type systems; compiler design; software tools putting theory into practice; cyber-physical, embedded, and real-time systems; distributed and concurrent systems; etc.

July 16-20 — 25th IEEE **International Conference on Software Quality, Reliability and Security** (QRS'2025), Hangzhou, China. Topics include: reliability, security, availability, and safety of software systems; software testing, verification, and validation; program debugging and comprehension; fault tolerance for software reliability improvement; modeling, prediction, simulation, and evaluation; metrics, measurements, and analysis; software vulnerabilities; formal methods; operating system security and reliability; benchmark, tools, industrial applications, and empirical studies; etc.

August 06-07 1st **Reliability and Maintainability Symposium European Edition** (RAMS Europe'2025), Amsterdam, the Netherlands. Deadline for registration: July 15, 2025 (early).

☺ August 20-22 31st IEEE **International Conference on Embedded and Real-Time Computing Systems and Applications** (RTCSA'2025), Singapore. Topics include: real-time scheduling, timing analysis, formal methods for temporal guarantees, programming languages and run-time systems, applications and case studies of IoT and CPS, cyber-physical co-design, medical CPS, CPS software/system engineering, multi-core embedded system, compiler and embedded software, embedded system design tools and methodologies, fault tolerance, reliability and security, etc. Deadline for submissions: July 7, 2025 (tutorials). Deadline for early registration: July 18, 2025.

☺ August 25-29 31st **International European Conference on Parallel and Distributed Computing** (Euro-Par'2025). Dresden, Germany.

August 26 23rd **International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms** (HeteroPar'2025). Topics include: languages, libraries, and interfaces for heterogeneous parallel programming models; integration of parallel and distributed computing on heterogeneous systems; etc.

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.

August 25 32nd **International Workshop on Expressiveness in Concurrency** (EXPRESS'2025). Topics include: expressiveness and rigorous comparisons between programming languages and models (distributed, component-based, object-oriented, service-oriented); analysis techniques for concurrent systems; etc.

August 25-30 23rd **International Conference on Formal Modeling and Analysis of Timed Systems** (FORMATS'2025), Aarhus, Denmark. Co-located with CONCUR and FMICS as part of CONFEST 2025. Topics include: models and metrics for the correctness, performance, reliability, safety, and security of systems; languages and methods for the specification of quantitative properties of systems; techniques, algorithms, and data structures for analysis, evaluation, and verification of the above models, e.g., for model checking, testing, constraint solving, scheduling, optimization, and worst-case execution time analysis; etc.

September 03-05 18th **International Conference on the Quality of Information and Communications Technology** (QUATIC'2025), Lisbon, Portugal. Topics include: all quality aspects in ICT systems engineering and management, such as on verification, validation, and testing, evolution and maintenance, security and privacy, etc.

September 07-12 41st IEEE **International Conference on Software Maintenance and Evolution** (ICSME'2025). Auckland, New Zealand. Topics include: change and defect management, continuous integration/deployment, empirical studies of software maintenance and evolution, maintenance and evolution of model-based methods, maintenance versus release process, mining software repositories, productivity of software engineers during maintenance and evolution, release engineering, reverse engineering and re-engineering, software and system comprehension, software migration and renovation, software quality assessment, software refactoring and restructuring, software testing theory and practice, source code analysis and manipulation, technical debt, etc. Deadline for submissions: July 15, 2025 (student volunteers).

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, defense, emergency & rescue; logistics, industrial automation, off-shore technology; education & training.

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.

September 10-12 28th **Forum on specification & Design Languages** (FDL'2025), St. Goar, Germany. Topics include: results, experiences, advances, and new trends related to languages, tools, and techniques for developing software and hardware systems and combinations thereof; targeted systems encompass cyber-physical systems, distributed systems, real-time systems, embedded systems, mechatronics, IoT, and reactive systems; four non-limiting scientific areas of languages, semantics, verification and analysis, simulation. Deadline for submissions: July 19, 2025 (PhD Forum).

September 15-19 25th **International Conference on Runtime Verification** (RV'2025), Graz, Austria. Topics include: monitoring and analysis of runtime behavior of software, hardware, and cyber-physical systems; program instrumentation; logging, recording, and replay; combination of static and dynamic analysis; monitoring techniques for concurrent and distributed systems; fault localization, containment, resilience, recovery, and repair; etc.

Sep 28 – Oct 02 20th **International Conference on Software Engineering Advances** (ICSEA'2025), Lisbon, Portugal. Topics include: trends and achievements; advances in fundamentals for software development; advanced mechanisms for software development; advanced design tools for developing software; software performance; software security, privacy, safeness; advances in software testing; specialized software advanced applications; open source software; agile and lean approaches in software engineering; software deployment and maintenance; software engineering techniques, metrics, and formalisms; software economics, adoption, and education; etc.

Sep 28 – Oct 03 **Embedded Systems Week 2025** (ESWEEK'2025), Taipei, Taiwan. Includes CASES'2025 (International Conference on Compilers, Architectures, and Synthesis for Embedded Systems), CODES+ISSS'2025 (International Conference on Hardware/Software Codesign and System Synthesis), EMSOFT'2025 (International Conference on Embedded Software). Deadline for submissions: August 15, 2025 (Embedded System Software Competition).

October 02-03 23rd ACM/IEEE **International Symposium on Formal Methods and Models for System** Design (MEMOCODE'2025). Topics include: formal methods in system design that address the foundations, engineering methods, tools, or experimental case studies; modeling languages, methods, and tools (programming languages and models, software and system modeling languages, architecture and high-level hardware description languages, ...), formal methods and tools (correct-by-construction methods; contract-based design and verification; static, dynamic, and type theoretic analysis; verification; validation; test generation; ...), models and methods for developing critical systems (security-critical and safety-critical systems, cyber-physical systems, autonomous systems, ...), formal methods/models in practice, etc.

Sep 29 – Oct 02 44th **International Symposium on Reliable Distributed Systems** (SRDS'2025). Porto, Portugal. Topics include: distributed systems design, dependability, security, safety-critical distributed systems, etc. Deadline for submissions: August 1, 2025 (PhD Forum).

Sep 30 – Oct 01 9th **International Conference on Engineering Computer Based Systems** (ECBS'2025), Ilmenau, Germany. Theme: "Engineering of Complex Hard- and Software-Systems". Topics include: applying scientific principles to design, develop, verify, evaluate, and maintain software, while coping with complexity, helping to avoid omissions and invalid assumptions, managing real world changing issues, and producing the most efficient, economic and robust solutions.

October 01-02 2025 **International Conference on Software Engineering Research & Development** (SERD'2025), Oklahoma City, Oklahoma, USA & Online. Topics include: general and social aspects of software

engineering (SE); software design, testing, evolution, and maintenance; formal methods and theoretical foundations; programming languages (PLs), systems, and environments; object-oriented (OO) design and analysis; emerging SE technologies and dependability; distribution, componentization, and collaboration; concurrent, parallel, and distributed systems; etc. Deadline for submissions: July 15, 2025.

October 06-10 25th **International Conference on Formal Methods in Computer-Aided Design** (FMCAD'2025). Menlo Park, California, USA. Topics include: formal aspects of computer-aided system design, including verification, specification, synthesis, and testing; groundbreaking methods, technologies, theoretical results, and tools for reasoning formally about computing systems; etc.

☺ 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).

October 13-15 32nd **Static Analysis Symposium** (SAS'2025). Singapore. Co-located with SPLASH'2025. Topics include: static analysis as fundamental tool for program verification, bug detection, compiler optimization, program understanding, and software maintenance.

October 21-24 36th IEEE **International Symposium on Software Reliability Engineering** (ISSRE'2025), São Paulo, Brazil. Topics include: development, analysis methods and models throughout the software development lifecycle; dependability attributes (i.e., security, safety, maintainability, survivability, resilience, robustness) impacting software reliability; reliability threats, i.e. faults (defects, bugs, etc.), errors, failures; reliability means (fault prevention, fault removal, fault tolerance, fault forecasting); software testing and formal methods; software fault localization, debugging, root-cause analysis; reliability of AI-based systems; reliability of model-based and auto-generated software; reliability of open-source software; normative/regulatory/ethical spaces about software reliability; societal aspects of software reliability; etc. Deadline for submissions: July 5, 2025 (industry track papers), July 21, 2025 (workshop papers).

October 27-30 23rd **Asian Symposium on Programming Languages and Systems** (APLAS'2025), Bangalore, India. Topics include: programming paradigms and styles (object-oriented programming, programming languages for systems code, ...), methods and tools to specify and reason about programs and languages (programming techniques, type systems, static and dynamic program analysis, language-based security, model checking, testing, ...), programming language foundations, methods and tools for implementation (compilers, program transformations, refactoring, intermediate languages, run-time environments, garbage collection and memory management, build systems, ...), concurrency and distribution (concurrency theory, parallel programming, distributed computing, verification and testing of concurrent and distributed systems, ...), applications and emerging topics (programming languages and PL methods in education, security, privacy, signal processing, computer-aided design, ...; case studies in program analysis and verification). Deadline for submissions: August 23, 2025 (non-SRC poster abstracts), October 10, 2025 (non-SRC posters).

October 27-31 23rd **International Symposium on Automated Technology for Verification and Analysis (**ATVA'2025). Bangalore, India. Topics include: theoretical and practical aspects of automated analysis, synthesis, and verification of hardware and software systems; program analysis and software verification; analytical techniques for safety, security, and dependability; testing and runtime analysis based on verification technology; analysis and verification of parallel and concurrent systems; verification in industrial practice; applications and case studies of verification; automated tool support for verification; etc.

November 05-07 18th **International Conference on Verification and Evaluation of Computer and Communication Systems** (VECoS'2025), Paris, France. Topics include: analysis of computer and communication systems, where functional and extra-functional properties are inter-related; cross-fertilization between various formal verification and evaluation approaches, methods and techniques, especially those developed for concurrent and distributed hardware/software systems. Deadline for submissions: July 25, 2025.

☺ 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: August 14, 2025 (3rd round).

November 10-14 23rd **International Conference on Software Engineering and Formal Methods** (SEFM'2025), Toledo, Spain. Topics include: aspects of software engineering and formal methods; software development methods (formal modelling, specification, and design; software evolution, maintenance, re-engineering, and reuse); design principles (programming languages; abstraction and refinement; ...); software testing, validation, and verification (model checking, theorem proving, ...; testing and runtime verification; other light-weight and scalable formal methods; ...); security and safety (security, privacy, and trust; safety-critical, fault-tolerant, and secure systems; software certification); applications and technology transfer (component, object, multi-agent systems; real-time, hybrid, and cyber-physical systems; intelligent systems and machine learning; education; ...); case studies, best practices, and experience reports.

☺ November 13 **High Integrity Software Conference** (HISC'2025), Newport, South Wales, UK. Topics include: The need for trustworthy software across a broader range of industries.

☺ November 16 **Parallel Applications Workshop, Alternatives to MPI+X** (PAW-ATM'2025), St. Louis, Missouri, USA. Co-located with SC'2025. Topics include: alternatives to the MPI+X model, novel application development using high-level parallel programming languages and frameworks; examples that demonstrate performance, compiler optimization, error checking, and reduced software complexity; experience with the use of new compilers and runtime environments; etc. Deadline for submissions: July 24, 2025.

November 19-21 20th **International Conference on integrated Formal Methods** (iFM'2025), Paris, France. Topics include: recent research advances in the development of integrated approaches to formal modelling and analysis; all aspects of the design of integrated techniques, including language design, verification and validation, automated tool support and the use of such techniques in software engineering practice. Deadline for submissions: August 15, 2025 (artifact registration), August 22, 2025 (artifact submission).

☺ November 26-28 6th **International Conference on Reliability, Safety and Security of Railway Systems** (RSSRail'2025), Pisa, Italy. Topics include: safety in development processes and safety management; combined approaches to safety and security; system and software safety analysis; formal modelling and verification techniques; system reliability; validation according to the standards; tool and model integration, tool chain; domain-specific languages and modelling frameworks; model reuse for reliability, safety and security; etc. Deadline for submissions: September 26, 2025 (posters).

December 01-03 11th **International Symposium on Dependable Software Engineering: Theories, Tools and Applications** (SETTA'2025). Oxford, UK. Topics include: formalisms for modeling, design and implementation; model checking, theorem proving, and decision procedures; scalable approaches to formal system analysis; formal approaches to simulation, run-time verification, and testing; integration of formal methods into software engineering practice; contract-based engineering of components, systems, and systems of systems; formal and engineering aspects of software evolution and maintenance; parallel and multicore programming; embedded, real-time, hybrid, probabilistic, and cyber-physical systems; mixed-critical applications and systems; safety, reliability, robustness, and fault-tolerance; dependability of smart software and systems; applications and industrial experience reports; software tools to assist the construction or analysis of software systems; etc. Deadline for submissions: August 20, 2025 (papers).

☺ Dec 02-05 46th IEEE **Real-Time Systems Symposium** (RTSS'2025), Boston, Massachusetts, USA. Topics include: innovations covering all aspects of real-time systems, including theory, design, analysis, implementation, verification, evaluation, and experience. Deadline for submissions: July 15, 2025 (Hot Topics Day proposals).

December 03-05 28th **Brazilian Symposium on Formal Methods** (SBMF'2025). Recife, Pernambuco, Brazil. Topics include: development, dissemination, and use of formal methods for construction of high-quality computational systems; applications of formal methods to software design, software development, software code generation, software testing, software maintenance, evolution and reuse, ...; specification and modeling languages (logic and semantics for specification and programming languages; formal methods for timed, real-time, hybrid, or safety-critical systems; formal methods for cyber-physical systems; ...); verification and validation (abstraction, modularization, and refinement techniques; static analysis; model checking; theorem proving; software certification; correctness by construction);

experience reports (reports on teaching formal methods, reports on the industrial application of formal methods); etc. Deadline for submissions: August 1, 2025 (paper registration), August 8, 2025 (papers).

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

2026

☺ January 11-17 53rd ACM SIGPLAN **Symposium on Principles of Programming Languages** (POPL'2026). Rennes, France. Topics include: all aspects of programming languages and programming systems, both theoretical and practical, ranging from formal frameworks to experience reports; principled, enduring contributions to the theory, design, understanding, implementation, or application of programming languages. Deadline for submissions: July 10, 2025.

January 12-13 **International Conference on Certified Programs and Proofs** (CPP'2026). 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; verification of correctness and security properties; teaching mathematics and computer science with proof assistants; etc. Deadline for submissions: September 5, 2025 (abstracts), September 12, 2025 (papers).

January 12-13 27th **International Conference on Verification, Model Checking, and Abstract Interpretation** (VMCAI'2026). Rennes, France. Co-located with POPL'2026. Deadline for submissions: September 10, 2025 (papers).

Jan 31 – Feb 04 IEEE/ACM **International Symposium on Code Generation and Optimization** (CGO'2026). Sydney, Australia. Topics include: a wide range of optimization and code generation techniques and related issues at the interface of hardware and software. Deadlines for paper submissions: September 11, 2025 (2nd round).

March 17-20 33rd IEEE **International Conference on Software Analysis, Evolution, and Reengineering** (SANER'2026). Limassol, Cyprus. Topics include: software analysis, parsing, and fact extraction; software maintenance and evolution, evolution analysis; software reverse engineering and reengineering; program comprehension; software architecture recovery and reverse architecting; program transformation and refactoring; mining software repositories and software analytics; software reconstruction and migration; program repair; software release engineering, continuous integration and delivery; software tools for software evolution and maintenance; empirical studies on all the above topics; education related to all of the above topics. etc. Deadline for submissions: August 20, 2025 (workshops), October 9, 2025 (research track abstracts), October 16, 2025 (research track papers), November 7, 2025 (registered report track), November 10, 2025 (industrial track abstracts, short papers and posters track abstracts, Early Research Achievement (ERA) track abstracts, reproducibility studies and negative results (RENE) track abstracts), November 17, 2025 (industrial track papers, short papers and posters track papers, Early Research Achievement (ERA) track papers, reproducibility studies and negative results (RENE) track papers, tool demo track papers), December 8, 2025 (journal-first track papers), December 12, 2025 (workshop abstracts), December 18, 2025 (workshop papers), January 12, 2026 (tutorials).

April 11-16 29th ETAPS **International Joint Conferences on Theory and Practice of Software** (ETAPS'2026), Torino, Italy. Deadline for submissions: July 26, 2025 (satellite events), October 16, 2025 (ESOP round 2, TACAS, FoSSaCS, FASE), October 30, 2025 (TACAS artifacts), January 8, 2026 (non-TACAS artifacts).

April 11-14 35th **European Symposium on Programming** (ESOP'2026) 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. Deadline for submissions: October 16, 2025 (round 2).

April 11-16 29th **International Conference on Foundations of Software Science and Computation Structures** (FoSSaCS'2026). Topics include: theories and methods to

support analysis, integration, synthesis, transformation, and verification of programs and software systems; language theory, automata, and games; type theory and proof theory; concurrency theory and process calculi; semantics of programming languages; program analysis, correctness, transformation, and verification; logics of programming; software specification and refinement; models of software security; etc. Deadline for submissions: October 12, 2025 (papers), January 4, 2026 (artifacts).

April 12-18 — 48th IEEE/ACM **International Conference on Software Engineering** (ICSE'2026), Rio de Janeiro, Brazil. Topics include: mining software repositories; 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; evolution and maintenance; software reuse; refactoring; environments and software development tools; focusing on programming languages, environments, and tools supporting individuals, teams, communities, and companies; systems and software traceability; 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; program analysis; debugging and fault localization; runtime analysis and/or error recovery; etc. Deadline for submissions: July 18, 2025 (research track papers), October 20, 2025 (workshop papers).

May 20-22 — 27th **International Symposium on Formal Methods** (FM'2026), Tokyo, Japan. Topics include: development and application of formal methods in a wide range of domains including trustworthy AI, computer-based systems, systems-of-systems, cyber-physical systems, security, human-computer interaction, manufacturing, sustainability, energy, transport, smart cities, healthcare and biology; techniques, tools, and experiences in interdisciplinary settings; experiences of applying formal methods in industrial settings; design and validation of formal method tools; etc. Deadline for submissions: November 25, 2025 (abstracts), December 2, 2025 (full papers).

♦ June 09-12 — 30th **Ada-Europe International Conference on Reliable Software Technologies** (AEiC'2026). Västerås, Sweden. Organized by Ada-Europe and Mälardalen University (MDU). Deadline for submissions: January 16, 2026 (journal track papers), February 27, 2026 (regular and industrial track papers, work-in-progress papers, tutorial and workshop proposals). #AEiC2026 #AdaEurope #AdaProgramming

June 22-26 — 23rd IEEE **International Conference on Software Architecture** (ICSA'2026). Amsterdam, the Netherlands. Deadline for submissions: October 3, 2025 (workshops), November 28, 2025 (Research Track paper abstracts, Software Architecture in Practice Track paper abstracts), December 5, 2025 (Research Track papers, Software Architecture in Practice Track papers).

☺ December — 47th IEEE **Real-Time Systems Symposium** (RTSS'2026). Asia, Africa, Oceania, or South America. Deadline for submissions: July 25, 2025 (hosting proposals).

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



General Chairs

Cristina Seceleanu, *Mälardalen University*
cristina.seceleanu@mdu.se

Kristina Lundqvist, *Mälardalen University*
kristina.lundqvist@mdu.se

Journal track Co-chairs

Axel Jantsch, *TU Wien*
axel.jantsch@tuwien.ac.at

António Casimiro, *University of Lisbon*
casim@ciencias.ulisboa.pt

Industrial track Co-chairs

Sahar Tahvili, *Einride AB*
sahar.tahvili@einride.tech

Rong Gu, *Mälardalen University*
rong.gu@mdu.se

Regular papers track Co-chairs

Antonio Fillieri, *Amazon Web Services*
afilieri@amazon.com

Peter Backeman, *Mälardalen University*
peter.backeman@mdu.se

Work-in-progress track Co-chairs

Malvina Latifaj, *Mälardalen University*
malvina.latifaj@mdu.se

Federico Aromolo, *Scuola Sant'Anna*
federico.aromolo@santannapisa.it

Workshops Chair

Dene Brown, *SysAda Limited*
dene.brown@sysada.co.uk

Tutorial Chair

Henry Muccini, *University of L'Aquila*
henry.muccini@univaq.it

Exhibition & Sponsorship Chair

Tiberiu Seceleanu, *Mälardalen University*
tiberiu.seceleanu@mdu.se

Finance Chair

Ahlan Marriott, *White Elephant GmbH*
ahlan@ada-switzerland.ch

Publicity Chair

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

Webmasters

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

Leo Hatvani, *Mälardalen University*
leo.hatvani@mdu.se

Local Chair

Susanne Frønnå, *Mälardalen University*
susanne.fronna@mdu.se

General Information

The **30th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2026)** will take place in Västerås, Sweden. The conference schedule comprises a journal track, an industrial track, a regular papers track, a work-in-progress track, a vendor exhibition, parallel tutorials, and satellite workshops.

- Journal track papers present journal-level 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.
- Regular track papers present conference-level (full paper) research advances.
- 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

16 January 2026	Deadline for submission of journal track papers
27 February 2026	Deadline for submission of industrial track papers, regular track papers, work-in-progress papers, and tutorial and workshop proposals
27 March 2026	First round notification for journal track papers, and notification of acceptance for all other types of submissions
09-12 June 2026	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 methods, testing, and model-based engineering of critical systems
- High-integrity systems and reliability
- AI for safety-critical and high-integrity systems engineering
- Real-time systems
- Ada language and technologies
- Applications and experimental evaluations in relevant domains

More specific topics are described on the conference web page.

Call for Journal Track Submissions

The conference includes a journal track, which seeks original and high-quality papers, not submitted elsewhere, which describe mature research work on the conference topics. Accepted journal track papers will be published in a Special Issue of Elsevier's JSA (https://www.sciencedirect.com/journal/journal-of-systems-architecture/publish/guide-for-authors) – the *Journal of Systems Architecture* (Q1 ranked, CiteScore 8.5, impact factor 3.7). Contributions must be submitted by **16 January 2026** and should not exceed **25 pages** (in case the authors need more pages the journal track chairs need to be contacted)**.**

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 (including major revisions) will be invited to present their work at the conference. Prospective authors may direct all enquiries regarding this track to the corresponding chairs.

Call for Regular Track Submissions

The conference includes a regular track, which seeks original and high-quality submissions, not submitted elsewhere, which describe mature research work on the conference topics. Accepted papers will be published in the *Open Access Series in Informatics* (OASICS series: https://drops.dagstuhl.de/entities/series/OASIcs, approval pending) and should be formatted according to the one-column format, not exceeding **16 pages**, excluding references. Contributions must be submitted by **27 February 2026.**

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 **27 February 2026**.

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* (https://www.ada-europe.org/auj/scope/), which will form the proceedings of the industrial track of the Conference.

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 or regular tracks. 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 M.Sc. graduates and Ph.D. students are especially sought. Authors shall submit their work by **27 February 2026**.

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 the 1-pagers) for publication in the *Ada User Journal* (https://www.ada-europe.org/auj/scope/), which will form the proceedings of the WiP track of the conference.

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 **27 February 2026** to the respective chairs. 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 **27 February 2026** to the respective chair. Once submitted, each workshop proposal will be evaluated by the conference organizers as soon as possible.

Venue



The conference will take place at Mälardalen University, Västerås, Sweden. Mälardalen University (MDU), one of Sweden's youngest and most dynamic universities, is a leading institution for applied research and innovation in collaboration with industry and the public sector. Established in 1977 and granted full university status in 2022, MDU is home to over 17,000 students and 1,000 staff members, with strong international ties and research excellence in areas such as embedded systems, future energy, health and welfare, and sustainable development. The university houses several internationally recognized research environments, out of which Embedded Systems, Innovation and Product Realization, and Future Energy are among the most prominent ones. MDU's campuses (Västerås and Eskilstuna) are modern, centrally located, and purpose-built to foster interdisciplinary collaboration and innovation.

Västerås is one of Sweden's oldest cities and now a hub for advanced technology, energy innovation, and digital transformation. Strategically located just an hour west of Stockholm and easily accessible by train, bus or car, Västerås is home to global tech companies such as ABB, Alstom, Westinghouse. The city sits on the shores of Lake Mälaren and combines scenic beauty, a vibrant cultural life, and excellent infrastructure. Västerås Airport offers direct flights to several European destinations, and Stockholm Arlanda International Airport is reachable within 90 minutes. With its blend of historical charm, green parks, and cutting-edge industry, Västerås offers the perfect setting for an inspiring and high-impact academic gathering.

LLM-based Framework for Email Classification and Phishing Detection

Nuno Costa, José Cecílio

LASIGE, Faculdade de Ciências da Universidade de Lisboa, Portugal; email: nuno.f.costa@edu.ulisboa.pt, jmcecilio@ciencias.ulisboa.pt

Ruben Salgueiro, Mauricio Rosa, Dulce Domingos

Reitoria da Universidade de Lisboa, Universidade de Lisboa, Portugal; email: {rsalgueiro, mauricio.rosa, mddomingos}@reitoria.ulisboa.pt

Abstract

Phishing and spam attacks continue to pose significant security risks due to their evolving nature, with traditional email classification methods often unable to effectively address these challenges. This paper proposes a framework that uses Large Language Models (LLMs) for enhanced email categorization and phishing detection. Through an inbox plugin, a user reports an email, which is then stored in a database, cleaned, and thoroughly analyzed using a series of specific prompts determining a classification of "normal", "unwanted", or "malicious" email.

Keywords: Email classification, phishing detection, spam filtering, Large Language Models (LLMs), cybersecurity.

1 Introduction

The volume of emails received daily has increased significantly, both in corporate and personal environments [1]. This scenario highlights the need for effective methods to categorize emails and mitigate security risks. However, people struggle to identify legitimate emails and this uncertainty can lead to unauthorized access and compromise of sensitive information [2], as attackers and spammers continuously refine their strategies.

Email categorization, a crucial part of email security, is the process of automatically sorting incoming emails into predefined categories based on their content, sender, metadata, and other characteristics. This process has various approaches that can be broadly categorized into non-intelligent and intelligent methods, each with their own strengths and limitations.

Non-intelligent methods provide a fundamental layer of security but often struggle to adapt to evolving threats:

- Blacklists: Depend on frequent updates.
- Email Authentication Protocols (SPF, DKIM, DMARC): Frequently misconfigured, allowing attacks.
- Heuristic-Based Detection: Uses patterns indicative of phishing or spam but can produce false positives, impacting legitimate emails [3].
- User Education & Awareness Training: Requires ongoing effort and is prone to human error.

Intelligent methods complement the detection but are not without limitations:

- Content-Based Analysis: Attackers use AI-generated content and obfuscation techniques to evade detection [4, 5].
- Real-Time Sandboxing & Threat Intelligence: While effective, some sophisticated attacks manage to conceal their malicious intent from automated analysis tools.

A robust phishing and spam detection system must integrate multiple techniques to maximize accuracy and minimize vulnerabilities. State-of-the-art phishing detection mainly relies on supervised learning algorithms, including Support Vector Machines (SVM), Logistic Regression (LR), Decision Trees (DT), and Naïve Bayes (NB) [6].

However, a fundamental limitation of current detection systems is their fragmented approach—each method analyzes only specific components of an email (e.g., sender identity, headers, textual content) in isolation. This allows adversaries to craft emails that bypass each security layer individually, despite the message being clearly fraudulent.

Jiang [7] demonstrated GPT-3.5 and GPT-4's effectiveness in detecting phishing elements in emails, while the ChatSpamDetectors system achieved 99.7% accuracy using GPT-4 [8]. Despite their potential, LLM-based approaches face challenges such as bias, privacy concerns, and limited dataset sizes, constraining their ability to generalize phishing patterns.

This work proposes an email categorization framework based on Large Language Models (LLMs) to address the limitations of existing security mechanisms. The role of LLMs in cybersecurity is an emerging field [9, 10]. Studies highlight their ability to analyze complex language patterns, identify vulnerabilities, and support cybersecurity professionals [11, 12].

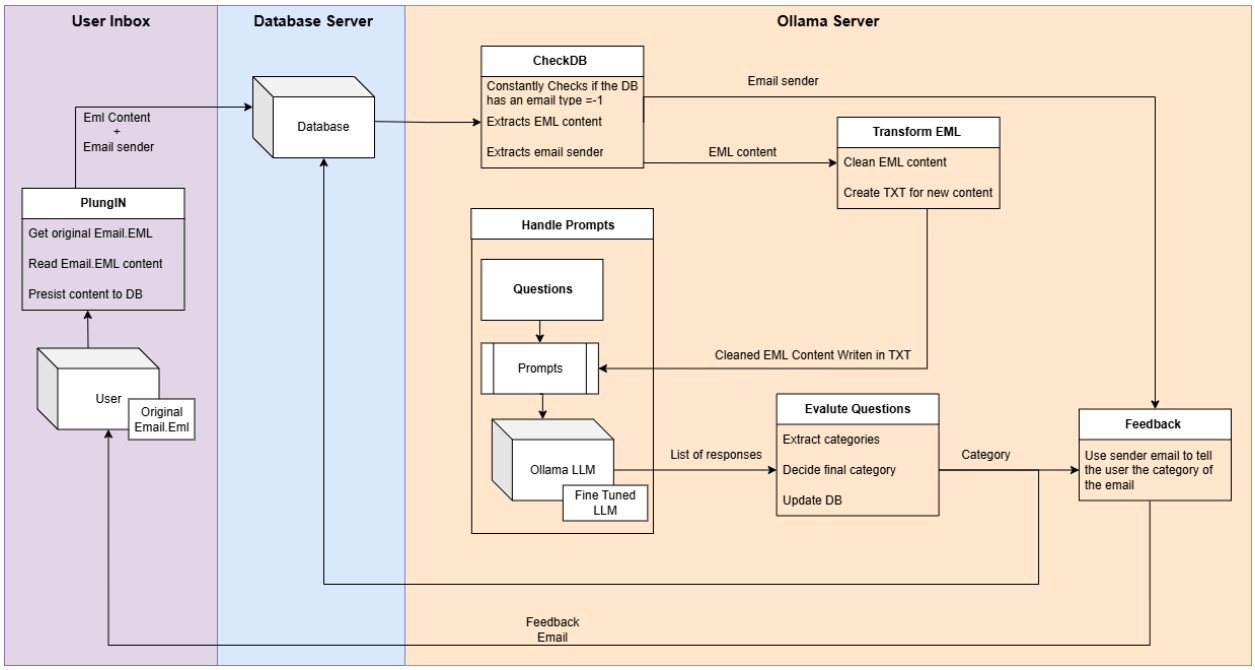


Figure 1: Email categorization framework

Unlike traditional methods that analyze discrete components of an email separately, our approach processes the entire email holistically. The system directly ingests the raw email source code, which includes: Headers (sender authentication, routing information, metadata), Content (textual information, embedded links, attachments) and HTML/CSS code (structured email design and hidden elements). By incorporating textual and structural email features, the framework aims to identify subtle phishing indicators that traditional machine-learning classifiers may overlook.

2 The categorization framework

Figure 1 shows the proposed framework for email categorization. The categorization process begins in the user inbox, where a plugin button allows users to report emails for analysis. These emails are subsequently stored in a database server. On the Ollama server, a script retrieves the stored emails, filters out non-essential content, and uses the remaining information as context for a series of prompts. The system then generates multiple prompts with targeted questions to determine the email category. Based on the responses obtained, the system classifies the email, updates the database accordingly, and provides feedback to the user.

This framework comprises three main components: (1) **User Inbox**, which corresponds to the Inbox where users report emails for analysis, (2) **Database Server** used to store the reported emails and track their classification status and (3) **Ollama Server** which processes and classifies the emails using a series of specific prompts.

2.1 User Inbox

A plugin featuring a "Report" button must be installed by the user in their inbox[1]. It interacts with the EML file format, which stores the complete content of an email, including headers, body, attachments, and metadata, in plain text format following the MIME standard. When a user clicks the "Report" button, the plugin extracts the EML content from the open email. The extracted content and the user's email address are then stored as a new entry in the database server. The system subsequently analyzes the email, and the user receives a feedback email indicating the determined category of the reported email. Currently, the process is not fully automated. It relies on users to identify potentially malicious or unwanted emails and report them for analysis.

2.2 Database Server

The database server hosts a structured database containing a single table named "Emails". This table stores reported emails, with each entry including a unique ID, timestamp, EML content, sender information, and an email classification type.

When a user reports an email, a new entry is created in the table, initially assigned a type value of "-1," indicating that the email has not yet been analyzed. The Ollama server continuously monitors the database for entries with type = -1. Upon detection, the email is retrieved for analysis. Once classified, the email type is updated to one of three categories: "0" (*normal*), "1" (*unwanted*), or "2" (*malicious*).

This updated database can further be leveraged to extract sender information from unwanted and malicious emails, allowing the implementation of blacklist mechanisms to enhance security.

2.3 Ollama Server

The Ollama Server comprises five main components: CheckDB, TransformEML, HandlePrompts, EvaluateQuestions, and Feedback. These components support Ollama

[1]We have developed such a plugin for Outlook users in our implementation of the framework. However, similar plugins could easily be developed for other popular email clients.

Server functions to process and structure email inputs, interpret and retrieve relevant information, and continuously improve system performance through feedback components. This creates a seamless and innovative response system.

The **CheckDB** component continuously monitors the database for entries where the type is set to -1. Upon detecting such an entry, it retrieves the corresponding email record from the database. The CheckDB component then saves the EML content and forwards it to the TransformEML module for processing. Additionally, it sends the email sender's address (i.e., the address of the user who reported the email) to the Feedback component.

The **TransformEML** component is responsible for cleaning the original source code of the email by removing non-essential parts that do not contribute to email categorization. Specifically, it eliminates inline and external CSS, as well as Base64-encoded images, among other unnecessary elements. Once processed, the component generates a TXT file containing the cleaned EML content and forwards it to the HandlePrompts module.

The **HandlePrompts** component interacts with the LLM by generating and submitting a series of questions to assess the nature of the email. Examples of these questions include:

- Is this email normal, unwanted, or malicious?
- Based on the headers, is this email normal, unwanted, or malicious?
- Based on the links, is this email normal, unwanted, or malicious?
- Considering the overall content, is this email normal, unwanted, or malicious?

These questions are designed to categorize the email, with some focusing on specific elements (e.g., headers or links) and others analyzing the email in its entirety. The HandlePrompts component constructs a dedicated prompt for each question, incorporating the email as context. The responses are stored in an array and subsequently passed to the EvaluateQuestions component for further analysis.

The **EvaluateQuestions** component processes the responses by extracting the assigned category from each answer and storing them in an array. It then determines the final category based on the most frequently occurring classification within the array. Once the final decision is made, the component updates the email classification to one of three categories: "0" (normal), "1" (unwanted), or "2" (malicious). Finally, it sends the determined category to the Feedback component.

Future work may include assigning weights to different questions, as some may yield more accurate results than others. Additionally, users or security teams could be provided with a confidence score (e.g., "70% confidence that this email is phishing"). These confidence levels could be stored in the database, allowing for automatic blocking of email domains associated with high-confidence phishing or spam classifications.

The **Feedback** component utilizes the email received from the CheckDB component and the category determined by the EvaluateQuestions component to generate a feedback response. It then sends an email to the user who reported the message, informing them of the assigned category.

3 The Large Language Model

The Large Language Model (LLM) is the core of our proposed framework for email classification and phishing detection. It leverages advanced natural language processing to analyze and accurately categorize email content. The LLM is used to identify misleading language patterns, suspicious sender details, and embedded threats such as malicious links or attachments, helping to distinguish between legitimate communications and sophisticated phishing attempts. Figure 2 represents the information sources used to train the LLM used in our framework.

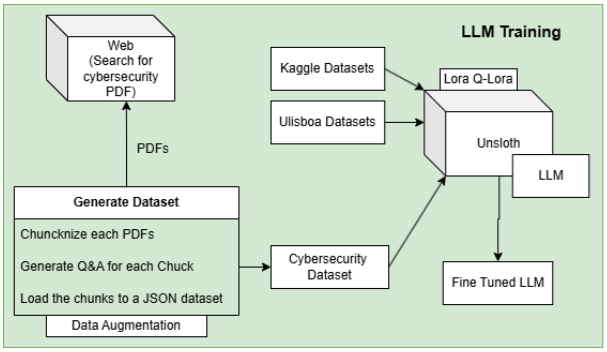


Figure 2: LLM Training

3.1 Choosing the Model

To determine the most suitable LLM for email classification, we first established key selection criteria:

1. Only open-source models were considered.
2. The model needed to be relatively lightweight, with a parameter range between 2 and 16 billion.
3. The model had to be compatible with Ollama.

These criteria significantly narrowed the selection, eliminating many mainstream models.

The remaining considered models were deployed locally, and their processing speed and classification accuracy were evaluated using a custom dataset. This dataset consisted of 19 emails, each manually labelled into one of three categories: "normal", "unwanted," or "malicious".

The final candidates, which demonstrated the best accuracy and relatively good speed, were Llama 3.1, Mistral-Nemo, Qwen 2.5, and PHI-4.

3.2 Training the Model

An important consideration in the development of the LLM was the choice of training methodology. We evaluated three approaches: full fine-tuning, fine-tuning with LoRA, and fine-tuning with QLoRA, considering metrics such as memory used, training speed, VRAM need, training cost, and performance/accuracy. Table 1 shows the results of our evaluation.

Feature	Full Tuning	LoRA	QLoRA
Memory Use	High	Low	Very Low
Speed	Slow	Fast	Faster
VRAM Need	Very High	Medium	Low
Training Cost	Expensive	Affordable	Very cheap
Performance	Best	Very good	Very good

Table 1: Comparison of Full Fine-Tuning, LoRA, and QLoRA

Ultimately, we opted for a training strategy primarily focused on LoRA and QLoRA due to their significantly higher efficiency, lower computational costs, and reduced hardware requirements. However, given the success of this framework, future work should explore full fine-tuning to further enhance the model's performance and adaptability. To help implement LoRA and QLoRA training, we chose the Unsloth platform, which provides a streamlined, user-friendly solution for efficient fine-tuning with minimal setup.

Finally, we re-evaluated the final candidates, the only models with current Unsloth support were Llama 3.1 and Qwen 2.5. Given the size of the email's source code, we selected Llama 3.1 due to its superior context length.

3.3 Training Datasets

Three primary sources of datasets will be utilized for training the LLM. The first source is Kaggle, which offers free datasets, including collections of labelled emails. The second source involves collaboration with the IT department of the University of Lisbon, which will provide a dataset of emails flagged as spam and/or phishing. The third and last dataset is currently being built and will be used to teach the LLM the basics of cybersecurity. It relies on a Generate Dataset component we have developed that can divide the documents into smaller chunks and then, using data-augmentation, can generate question-and-answer (Q&A) pairs for each chunk. To create the dataset, we will collect cybersecurity-related PDFs from trusted sources and feed them into this component; the consolidated Q&A pairs will constitute the dataset.

4 Final Remarks

This paper introduces an LLM-based framework for improved email classification and phishing detection using an inbox plugin, a database server, and the Ollama server for email analysis. While the framework shows promise in enhancing email classification, there is still room for improvement. The system currently relies on users to report each email, but automating the reporting process would enhance the system's usability and efficiency. The most frequently occurring category determines the final classification. However, not all questions contribute equally to classification accuracy. To address this, a weighting system should be implemented to refine the decision-making process. In conclusion, this LLM-based framework forms a good base for a system that can be enhanced in the future.

5 Acknowledgments

This work was supported by the University of Lisbon through an Initiation to Investigation Grant and by FCT through the LASIGE Research Unit, ref. UID/408/2025.

References

[1] E. H. Tusher, M. A. Ismail, M. A. Rahman, A. H. Alenezi, and M. Uddin, "Email spam: A comprehensive review of optimize detection methods, challenges, and open research problems," *IEEE Access*, vol. 12, pp. 143627–143657, 2024.

[2] L. Halgaš, I. Agrafiotis, and J. R. Nurse, "Catching the phish: Detecting phishing attacks using recurrent neural networks (rnns)," in *International Workshop on Information Security Applications*, pp. 219–233, Springer, 2019.

[3] B. B. Gupta and M. Sheng, *Machine Learning for Computer and Cyber Security*. CRC Press: Boca Raton, FL, USA, 2019.

[4] P. Bountakas and C. Xenakis, "Helphed: Hybrid ensemble learning phishing email detection," *Journal of network and computer applications*, vol. 210, p. 103545, 2023.

[5] P. Bountakas, K. Koutroumpouchos, and C. Xenakis, "A comparison of natural language processing and machine learning methods for phishing email detection," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pp. 1–12, 2021.

[6] P. Verma, A. Goyal, and Y. Gigras, "Email phishing: Text classification using natural language processing," *Computer Science and Information Technologies*, vol. 1, no. 1, pp. 1–12, 2020.

[7] L. Jiang, "Detecting scams using large language models," *arXiv preprint arXiv:2402.03147*, 2024.

[8] T. Koide, N. Fukushi, H. Nakano, and D. Chiba, "Chatspamdetector: Leveraging large language models for effective phishing email detection," *arXiv preprint arXiv:2402.18093*, 2024.

[9] A. Soltani, D. K. Nkashama, J. F. Masakuna, M. Frappier, P.-M. Tardif, and F. Kabanza, "Assessing language models for semantic textual similarity in cybersecurity," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 370–380, Springer, 2024.

[10] F. Trad and A. Chehab, "Large multimodal agents for accurate phishing detection with enhanced token optimization and cost reduction," in *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pp. 229–237, IEEE, 2024.

[11] Y. Chen, M. Cui, D. Wang, Y. Cao, P. Yang, B. Jiang, Z. Lu, and B. Liu, "A survey of large language models for cyber threat detection," *Computers & Security*, p. 104016, 2024.

[12] V. Tanksale, "Cyber threat hunting using large language models," in *International Congress on Information and Communication Technology*, pp. 629–641, Springer, 2024.

Archaeology for the Future - Ada on OpenVMS

Gérard Calliet

pia-sofer, vms_ada_alliance; 66 rue de la mascotte 17190 Saint Georges d'Oléron - Cheray ; France ;Tel: +33 (0) 670 716 179; email: contact@vmsadaall.org

Abstract

This paper presents a project to make Ada available on all VMS platforms. The project is put in perspective within the broader framework of sustainable development cycles. We elaborate a proposal to revisit upward compatibility as a mode of temporal portability. The methods and tools used help to rethink archaeology as an image of intelligence in programming activity. The paper also highlights the similarities between the destinies of the Ada and VMS ecosystems.

Keywords: Ada, VMS, OpenVMS, LLVM, Gnat-llvm, backward compatibility, portability, software conception

1 A time trap

In 2015, the author of this article found himself in a time trap. After a very long gestation period, the decision was taken to relaunch (with Ada) an urban transport control system on the same Operating System where it had been created (1993): OpenVMS.

But the decision, matured in 2013 and 2014 on the basis of support for Gnat Ada on OpenVMS (Itanium), came in 2015, the year in which AdaCore decided to stop supporting Gnat Ada on OpenVMS (Itanium). In the midst of a reinvestment drive, the weight of the past came to the fore.

So, we had to go back to basics and rebuild Gnat Ada for Itanium from the FSF source code.

2 Why OpenVMS?

You might wonder why we chose to go back to OpenVMS, an OS that is a priori on the way to obsolescence.

OpenVMS is an OS of original conception, founded by people from MIT. Similar to Ada: there are times in computer science when you can rationally synthesize a set of problems.

This specific OS was chosen for its qualities for demanding applications. For certain environments, it has stood the test of time. One essential quality that has made this possible: backwards compatibility. You can write an application for VAX, and it will live on Alpha; then on Itanium, and soon on x86 and the cloud, without any major transformation.

In 2013 OpenVMS was also caught in a time trap. HP decided to stop supporting it. But the force of the past meant that OpenVMS was taken over by a start-up, VMS Software Incorporated (VSI) [1] - which has now ported it to x86.

Collaborating with AdaLabs company, we succeeded in our gamble in 2016 [2].

The dual success that allowed us to avoid both time traps gave us food for thought. We needed to go further in terms of application, and we revisited the methodological elements that emerged during our work.

3 Ada and VMS through ages

3.1 Introducing the project

The history of Ada has had its twists and turns. It seems that Ada and OpenVMS share the ability to risk oblivion and rise from the ashes.

Observing these similarities in their destinies has led us to consolidate what was initially just a sketch. The current project is to offer Ada on OpenVMS across all the ages of VMS: VAX, Alpha, Itanium, x86 (all environments still 'up and running').

The dual success that allowed us to avoid both time traps gave us food for thought.

We are thus consolidating Ada and VMS and exploring a new technical and economic field. This new field is linked to the existence of new emergencies named under the concepts of sustainable development, sustainability, energy efficiency, and cyclic economics. This is a universal trend that has perhaps not yet been widely explored in the field of IT development.

We present the different stages of making a universal availability of Ada on VMS, whether already existing or implemented, awaiting refinement, or still in the early stages of development.

3.2 Vocabulary

To facilitate the rest of the presentation, let's review the concepts, events, systems, and tools involved.

VAX stands for Virtual Address eXtension. This is a 32-bit CISC architecture created by DEC in the mid-1970s. It gave rise to a range of computers that became widely distributed worldwide. Some early Unix systems were developed on VAX systems.

DEC stands for Digital Equipment Corporation, co-founded by Ken Olsen. It was created with the idea of opening up wider access to computing resources by launching lines of minicomputers, PDP (16-bit), then VAX (32-bit).

VMS stands for Virtual Memory System. This is the operating system most often associated with VAX machines,

to the point that the common name is VAX/VMS, which expresses the compatibility of hardware and software.

Alpha is the name of the new hardware architecture created in the early 1990s by DEC. It is a 64-bit RISC architecture. The VMS operating system was ported to Alpha. The name VMS was later changed to OpenVMS. This remains the official name for VMS to this day. However, as a paradigm of a certain computer culture, the name VMS is most commonly used.

Itanium is the name of a 64-bit EPIC (Explicit Parallel Instruction Computing) architecture created by several manufacturers, including DEC, Bull, and Intel, in the early 2000s. This architecture was supposed to revolutionize the field of hardware architecture, but it failed to achieve real commercial success.

After the commercial failure of DEC, which was acquired by Compaq and then by HP, VMS gradually lost significant market share. In 2013, HP decided to end its support.

VSI stands for VMS Software Inc. This startup was formed to acquire the rights to OpenVMS and has maintained the operating system ever since. Its major investment was to port VMS to the standard x86 architecture.

x86 is the name of an architecture that has survived the ages and is now widely used.

GNAT Ada is the name associated with the open-source Ada compiler and all the compiler-related components. It is based, at least initially, on the gcc project (GNU Compiler Collection). At the request of the US Air Force, this open-source compiler was developed with New York University. It is now supported by AdaCore. The GNAT Ada sources are deposited in the FSF (Free Software Foundation) repository.

gcc is an open-source project housing all compiler generation tools, front-end and back-end, multi-language and multi-target platforms. The project was initiated in 1987 by Richard Stallman and is maintained by a very large community.

LLVM initially stood for Low Level Virtual Machine. It is a project similar in its goals to gcc, initiated in 2000 at the University of Illinois. The notion of a virtual machine was confusing, and LLVM became an orphan initialism. The composition of the front-end, the back-end and the nature of Intermediate Representation (IR) have been rethought. The project is also open source, but under licensing modes considered more open for industrial uses.

Debian is a major distribution of the Linux operating system, the basis for other Linux distributions. Debian versions are given names, for example, "Trixie" for version 13, which is a recent version, and "Wheezy" for version 7, released in 2013.

SIMH is the name of a family of emulators for older hardware, including VAX. Initiated by Bob Subnick in 1993 to preserve the legacy of minicomputers. The SIMH community is still very active.

4 Steps

4.1 Keeping DEC Ada alive for VAX and Alpha

For VAX and Alpha, the only effort is to preserve what already exists. We have the excellent DEC Ada for OpenVMS [3], renowned in particular for its debugging.

The compiler documentation is extremely valuable, including an LRM (Language Reference Manual) that includes the universal LRM and is annotated with implementation details for VMS. The compiler only offers version 83 of Ada.

There are still applications developed in DEC Ada that run on VAX and Alpha environments. I still give ADA / VAX-VMS training courses for teams who have to maintain their platform until at least 2035. A recollection of these uses must be done.

The VAX/VMS and OpenVMS worlds on Alpha do not have the same industrial contexts. When VMS was taken over by VSI, only the rights for VMS on Alpha and VMS on Itanium, and future rights on other platforms, were acquired by VSI. As a result, VMS on Alpha and Itanium, as well as products such as the DEC Ada compiler, are maintained by VSI. But the VAX/VMS system is, in a sense, an orphan, with no maintenance from HP, the current owner.

So, DEC Ada on VAX is somehow orphan, whereas DEC Ada on Alpha is still supported by VSI.

For the actual hardware context, both VAX/VMS and VMS on Alpha currently still run on VAX or Alpha hardware, as well as on commercial or open-source emulation solutions. The open source SIMH solution is very often used for VAX environments.

A special effort must be made to maintain the archives for VAX. Some efforts have to be made to register the operating system sources as part of the Software Heritage project [4]. This type of operation requires negotiations with the current owner of the rights to VAX/VMS. Another project could be to rebuild from these sources, following the procedures used by DEC, the latest version of VAX/VMS, version 7.3, as well as products such as DEC Ada, to have a searchable reference version.

For DEC Ada on VAX and DEC Ada on Alpha, the effort essentially focuses on managing the reference archives. In this context, communities like the one surrounding SIMH are essential reinforcements.

4.2 Restarting GNAT ADA on OpenVMS Itanium

The Itanium adventure is more difficult. HP had abandoned Ada support to AdaCore, transition from DEC Ada to GNAT Ada on VMS/Itanium. And AdaCore stopped all support for Itanium. One of the reasons for this was the switch from gcc 4.7 to gcc 5, with the appearance of C++ in the tool chain. For AdaCore, the adaptation work for OpenVMS was not considered profitable, and we were in a period of disintegration of OpenVMS support by HP.

For the purposes of our application, we didn't need the latest GNAT version. The last (AdaCore) GNAT Ada version for

OpenVMS Itanium offered Ada version 83 and Ada version 95. The applications we were porting to Itanium needed only version 83, with some Ada 95-like additions that had been offered by DEC Ada alpha. We rebuilt the latest GNAT Ada version for OpenVMS Itanium, which is based on gcc version 4.7. We rebuilt it from FSF sources.

Archaeology: gcc 4.7 was no longer really supported by the gcc community. And we had to find the build procedures used by AdaCore ourselves. In this kind of situation, the engineer is an archaeologist: reconstructing the procedures from the traces, looking for existing repositories, adapting to older versions.

Our build is available on github [5]. We had to use the wheezy version of Linux, accessed via schroot. Schroot allows the use of another version of Linux on a Linux host. To rebuild a gcc version 4.7, you must be in the same conditions as its initial build, namely Debian 7, named wheezy. Again, this requires archive work, because the wheezy repositories are now in the archive domain. It is also necessary to find all the necessary components in their version consistent with gcc 4.7.

The build procedure is of the Canadian type. This procedure has three phases: creation of a native gcc 4.7 compiler, that is, one that runs under Linux (Debian 7 wheezy); creation with this native gcc 4.7 of a gcc 4.7 cross-compiler that creates VMS binaries by working on Linux; and finally, use of this compiler to create a compiler executable on VMS producing VMS executables. This type of approach is one of the possibilities offered by gcc.

It is important to note that the operations are performed with major use of the Linux make tool, part of the Debian tool chain, and these tools are themselves versioned and relative to the Debian version used. In our approach, it is therefore necessary to choose a tool chain consistent with what we are rebuilding. It must be chosen and, if necessary, found in archives.

We were thus able to reconstruct the latest version of GNAT Ada existing on Itanium, based on version 4.7 of gcc.

The underlying gcc 4.7, on top of which the GNAT Ada compiler exists, could be extended and used as a C compiler (gcc-like) under OpenVMS Itanium. This compiler could itself be used to facilitate ports of open-source projects to OpenVMS Itanium of a similar era. The gcc C idiom is probably closer to what is used in open-source projects than the DEC C idiom existing under OpenVMS. This type of extension, which goes beyond the Ada domain, demonstrates the existence of additional benefits of our archaeological approach.

The next step for Itanium is to take the first step towards gcc 5. We'll have to reinvent what AdaCore could have done then. With the same problem of the extreme age of gcc 5. The industrial interest of this operation is reduced, since the lifespan of Itanium platforms is itself reduced, and new development on Itanium with more recent GNAT Ada versions seems unlikely. However, the scientific effort of redevelopment using archaeological methods is interesting.

4.3 Move to x86

The current phase, closer to industrial opportunities, is the move to x86. Formally, the switch is entirely feasible. VSI has chosen the LLVM tool chain for its compilations of the OS and its environment.

Here too, it was a Canadian-style strategy that allowed VSI to create VMS compilers on x86[6]. VSI began its journey by cross-compiling from OpenVMS Itanium to x86. In the first phase, it was only possible to refer to version LLVM 3.4.2, due to the antiquity of the C and C++ compilers present on OpenVMS Itanium. In 2025, VSI implemented, using Canadian-style procedures, an LLVM and Clang environment in version 10.0.1 on VMS x86. Very soon, using these tools, it will be possible to natively generate a more recent LLVM environment, with the associated tool chain tools, essentially a compatible version of CMake.

Furthermore, AdaCore has initiated a GNAT LLVM project that allows the construction of the GNAT Ada compiler with an LLVM underlying framework. The GNAT LLVM project [7] and the OpenVMS LLVM tool chain therefore need to be brought together.

Here again, archaeology. For reasons of feasibility, VSI has chosen older versions of LLVM as the initial basis of its effort and will migrate at its own pace to more recent LLVM versions.

While waiting for recent versions of LLVM to be made available, we need to search the GNAT LLVM project for dependencies on recent versions of LLVM and find workarounds. This implies tracing (archaeology again) the methods used to adapt GNAT to LLVM. Initially, we are aiming for cross-compilation for execution on OpenVMS.

4.4 Learned by doing, elements of the method

A first characteristic of the method is the great importance attached to archives. It is absolutely necessary to be able to locate all sources for a given period, and to be able to determine which versions of the different elements can be used together.

In this effort, the relationship with the communities likely to know and maintain the archives is crucial. It is likely that the approach even requires reactivating some communities that have previously lain dormant.

The situation in this type of project seems to be a few steps behind what is generally available. It would be a mistake to try to immediately make up for this delay. It would be a never-ending race. On the contrary, we must accept taking several steps back and equip ourselves with the means to analyze the reasons for the transitions.

The step back consists of understanding the dynamics of the transitions, their necessity, and what, in the change, has been preserved as essential. The result of this type of research converges towards a rediscovery of the fundamentals at play appearing in the dynamics of change. Once these analyses are completed, we can assess what can be reconstructed, or how we can circumvent problems that cannot be resolved as they stand.

The essence of the approach can be seen as a rebound. A search in the past for initial meanings, and a rethinking based on these initial meanings. Here we find the meaning and etymology of the notion of archaeology, a rediscovered logic of the principle, arche and logos.

Why is this archaeology particularly possible with VMS and Ada? Because both VMS and Ada were themselves conceived as a moment of synthesis, that is, a rebound from the fundamentals.

5 Towards temporal portability and its communities

Our approach, with its archaeological method, opens up another conception of the temporal organization of IT developments. Time is much more than a headlong rush. What we are working on with Ada on VMS is a deepening of what has until now been called backward compatibility.

The designers of OpenVMS were clearly aware of the need for backward compatibility. They associated this with all the qualities of robustness and a coherent, rational approach. They were not alone in guaranteeing this. Then this quality generally faded, giving way to LTS or assurance, which turned an intrinsic quality into a luxury.

We all know that the flight to the ever-new, with no regard for the old, is reaching its energetic and material limits. What is being developed throughout industry in terms of reusability, the circular economy and recycling is bound to affect the IT world in the long term. The concept of ‘temporal portability’ is therefore going to take hold in IT.

This can be particularly relevant with Ada. The issue of portability has always been very important in Ada. Until now, it has been primarily designed according to the notion of space (from platform to platform). We are probably in a time where portability according to the notion of time is becoming important.

6 More than just economic issues

The project is also driven by theoretical consideration. No foundation is possible without consideration of the fundamentals. Scientific thinking is akin to archaeology: looking for patterns in the traces of reality or previous theories. From this point of view, a project like ours has the potential to attract the interest of the intellectual elite.

The birth of Ada can be seen - like that of OpenVMS - as a moment of synthesis, a coherent re-elaboration of issues that have emerged over time. It's like a pause, a look back, then a rebound. It is this type of approach that is once again possible when we compare the life cycles of Ada and OpenVMS.

From this point of view, our project is a revival of this kind of gesture, a precursor of new ways of looking at IT life cycles.

7 Join the project

All specific skills in OpenVMS and tool chain management, knowledge of older versions of gcc, knowledge of the GNAT LLVM project, knowledge of LLVM versions, Ada theorists, etc. are welcome.

Those who join us have a taste for researching archives and historical witnesses to developments. They are decipherers of the treasures of ingenuity of which the programs are traces. They know how to reinvent the future from the past.

References

[1] New company for OpenVMS: VMS Software Incorporated: www.vmssoftware.com

[2] Our version of Gnat Ada for OpenVMS / Itanium: www.vmsadaall.org

[3] DECAda: https://docs.vmssoftware.com/docs/ada_lrm.pdf

[4] Software Heritage: https://softwareheritage.org.

[5] D. Sauvage, G. Calliet on github.com: https://github.com/AdaLabs/gnat-vms

[6] J. Reagan, journey porting OpenVMS compilers to x86: https://llvm.org/devmtg/2017-10/slides/Reagan-Porting%20OpenVMS%20Using%20LLVM.pdf and https://www.youtube.com/watch?v=mClf5Q7Aw6c

[7] Gnat Ada / LLVM project (community): https://github.com/AdaCore/gnat-llvm

Development and Management of Digital Twins for Energy Monitoring Systems

Tiago Carvalho, Luis Miguel Pinho

Polytechnic Institute of Porto, School of Engineering & INESC TEC, Porto, Portugal; email: {tdc, lmp}@isep.ipp.pt

Pedro Paiva

Cleanwatts S.A, Coimbra, Portugal; email: ppaiva@cleanwatts.energy

Abstract

The Internet of Things (IoT) enables everyday objects to connect and communicate remotely, transforming areas such as smart homes and industrial automation. A key application is Energy Monitoring Systems (EMS), which track energy consumption and production within houses and communities. Digital Twins (DT) is a growing concept that allows the management and visualization of physical systems via an equivalent digital system.

Designing and integrating DTs remains complex and are usually provided as generic approaches that provide more complexity than desired by an expert in the area of EMS. In this paper we provide an approach that allows the design of EMS and the automatic deployment of a Digital Twin instance. The approach automatically connects a physical (or simulated) system to its Digital Shadow counterpart, and provides management and visualization support via a set of tools. We provide a demonstration of the development phase and how the tools are used to manage the deployed system.

Keywords: EMS, Design, Digital Twins.

1 Introduction

The modern digital era is now defined by the seamless integration of the physical and digital worlds. The two most important concepts that influence this integration are the Internet of Things (IoT) systems and the Digital Twins (DT) concept. IoT systems are networks of interconnected devices, including sensors, actuating devices, and network connectivity. These devices collect, send, and act on data, usually requiring minimal to none human intervention. These systems bring to the forefront the concept of "smart systems", that can use existing information, for instance, for better user-feedback or optimization purposes.

An example of IoT systems are Energy Monitoring Systems (EMS) [1, 2], which are system-of-systems that monitor both energy consumption and production within, for instance, a house, a building or a company [1].

DTs are virtual replicas of physical assets or systems. They provide a dynamic representation of the physical entity, integrating real-time data, or simulation models, to mimic the status and behavior of their physical counterparts. The management and visualization of IoT systems via a DT is fundamental for guaranteeing correct operation of the system, to study its historical data and even to exploit that data for optimizing the system.

Designing a DT can be a complex process. A DT must integrate and configure different tools, including design, runtime management, communication, data storage and visualization [3]. Many DT modeling approaches provide generic modeling concepts for IoT systems [4]. While very versatile and providing completeness for the design of any IoT system, the existing approaches require a learning curve from the developer of the system, for understanding both the modeling process and how it can be applied to their type of systems. Usually, these developers (or designers) are skilled in their own area of expertise. In particular, a designer of EMS has full understanding the concepts of these systems, and thus would benefit from a specialized process.

A domain-specific modeling approach is imperative in this case, allowing the developer to fully focus on its area of expertise, and thus improve the designers productivity and the overall system design viability. Similar to the modeling of IoT systems, the instantiation of DTs is provided with a generic approach, usually considering the concept of "Things" [5].

In this work we aim to provide a specialized approach for modeling and management of Digital Twins for Energy Monitoring Systems. The approach provides a Domain-specific Language (DSL) for modeling EMS-related entities, including the devices and the general system architecture. The framework compiles the designed EMS into a set of artifacts that, when deployed, initialize and connect a DT instance with a physical system. The DT instance includes not only mechanisms to manage the DT itself, but also the capabilities of a database that stores historical data regarding the system execution, and the visualization of that data via dashboards.

We demonstrate this work with a real-life use case of EMS, named Cloogy® EMS. We show how to design a simple EMS and how the generated artifacts are handled to use the DT. In its current status, the approach provides a Digital Shadow (DS) of the real world system.

This paper is structured as follows. Section 2 provides a brief description of what is an EMS. Section 3 describes the overall

EMS-DT framework. Section 4 gives details regarding the DSL and the compiler. Section 5 shows how the generated assets are used for the instantiation process. Section 6 draws some conclusions and describes future work.

2 EMS

To better explain what an EMS is, we give an introduction to EMS and describe one real-life use case to exemplify. An EMS is a distributed sensing and actuating IoT system designed to observe and manage energy production and/or consumption [1]. They provide access to real-time and historical energy data. A smart meters and sensors are deployed to collect real-time energy data, including electricity, and water and gas usage, providing insights regarding patterns and enabling the identification of optimization opportunities.

It is common to have a centralized IoT system, where all data processing, analysis, and decision-making occur within a single centralized platform or server. The IoT devices collect and transmit data to a central hub, or a hierarchy of hubs if the system is very dense or dispersed. The central hub then manages, processes, and stores the data, often using cloud computing resources for scalability and accessibility. The data is usually offloaded and analysed in dedicated systems.

Cloogy® [6] is an example of EMS - designed by CleanWatts S.A. company - that monitor and control residential energy. They consist of a set of distributed devices, including power clamps, smart plugs, multi-sensor devices, and hubs. A hub is responsible for remotely managing these devices, acquire measurements from the sensors and provide access to the aggregated data. The number of devices is arbitrary and limited to the hub capabilities. To circumvent the hub limitations, several hubs can be considered, each one working independently as an individual EMS.

The hub receives data collected from a wide range of sensors and transmits it to a server. The collected data includes energy consumption and any other measurement obtained from the devices communicating with the hub [6]. The periodic acquisition of measurements is imperative in these systems since the sensing devices do not store any data, and only replies with the latest sensor reading.

3 EMS-DT Overview

Our solution uses a DSL-based modeling framework especially designed for the design of EMS. The framework provides the capabilities of generating DT instances interconnecting with the physical counterpart. Currently, the approach provides a DS of the system. This means that the DT is able to reenact the real-life system, but it is currently unable to interact with the real system.

Figure 1 shows the designed approach. The main entry of this work is the specification of the EMS in a DSL representation (the '.ems' file on top of the figure). These will include a set of files that represent the entire real-system, including the definition of the devices (sensors, actuators, hubs, etc.) and the interconnection between them.

From the system designed in the DSL, the compiler creates a set of artifacts that automatically instantiate the DT and the

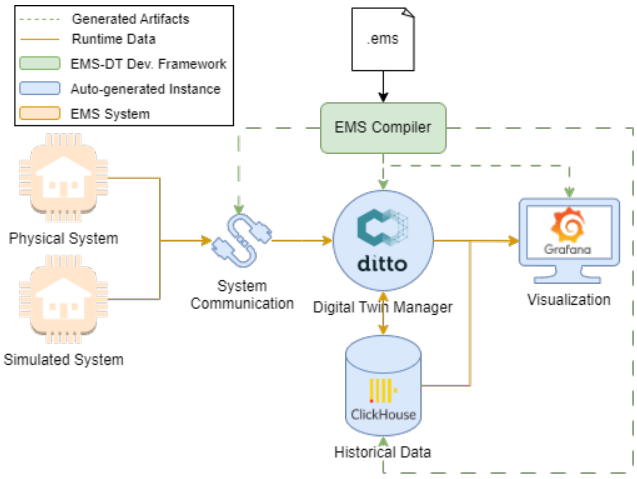


Figure 1: The EMT-DT framework, from the design of system to the instantiation of the DT.

connected features, including the setup for the database and the graphical representation of the data. When the system is deployed from the generated artifacts (the green dotted lines in Figure 1), three tools will be kept online: a digital twin management system, a database and a graphical user interface (GUI) for the visalization of the DT.

The instantiated DT manager communicates with the real system via a "System Communication", which is intended to be used as an interface that normalizes the communication between the real-world with the digital twin counterpart. Currently, the approach is considering only the communication via mqtt. Regardless, this interface will be improved to consider a more advanced approach, such as the use of Hono as the normalization framework. As shown in Figure 1, the real system can also consider the use of a simulation environment.

The real system keeps executing in a normal manner, and keeps feeding the DT with runtime data. When receiving new data from the real system, the DT updates its information and automatically sends the current information to the database to keep track of historical data.

This historical data is accessed via the "Visualization" Tool, which provides a set of dashboards, automatically generated for the system, that show the history of the behavior of the devices. For instance, in the case of sensors, the dashboard shows the acquired energy measurements along the time.

4 EMS DSL and Compilation

The DSL and the compiler where developed using a Xtext-based framework. The advantage being the automatic generation of essential assets - from the language grammar - for the development and compilation environment, since Xtext provides good integration with the Eclipse framework. The language follows a set of design rules to be more aligned with the expected standards of the "Web of Things" community.

Figure 2 shows an example of a sensing device designed with the DSL. The idea of the approach is: design once, instantiate as necessary. This means that a "device" designed in the DSL is, in fact, the "model" description of the device. That

```
1  using "units/Units"
2  device Plug:0.1
3     description = "This is a smart plug."
4     type = sensor
5     properties:
6        name: "Cloogy Plug"
7        model: "Cloogy(r) Plug"
8        manufacturer: "CleanWatts"
9     protocols: [Zigbee]
10    measurements:
11       InstCurrent: float unit=A
12          constraints:
13             acquisition_period: min=1 unit=s
14       InstVoltage: float unit=V <min=1 unit=s>
15       EnergyConsumed: i32 unit=Wh <min=1 unit=s>
```

Figure 2: Example of a sensor modeled in EMS-DT DSL.

description can then be used to create instances of a device, in other words, a real physical device. This approach can be seen as building "instances" from a class. For instance, Figure 2 shows a single plug device design, while the code in Figure 3 (in lines 5 and 11) uses the same "device" to create two different instances.

A device contains a textual description (line 3 of Figure 2, its type (e.g. sensor, actuator or hub), a set of static properties - information that rarely changes), and the protocols it supports. In the case of sensing devices, the device should also include a list of measurements that it can acquire.

Each measurement has a data type of the value to acquire, and a unit to describe the measurement. A set of constraints can also be set via the `constraints` attribute of a measurement. Line 13 of Figure 2 shows an example of a constraint for the measurement acquisition period, that indicates the minimum value for acquiring that measurement is one second. Lines 14 and 15 also demonstrate a shorthand for this particular property, which can be specified inline in a measurement via an encapsulation with the '<' '>' symbols. The design of other types of devices is very similar, especially for sensors.

Having the definition of the devices, it is possible to reuse these for modeling different system instances. Figure 3 is an example of a house designed with a set of pre-defined devices, which are "imported" via the `using` statement. A system is declared as as "system", and should contain four properties: the `devices` (line 4), the `network` (line 18), the DT/EMS `communication` and a set of `properties` (line 26).

The `devices` are a set of instances of devices, i.e. an instance of a real device, to which we can provide an identification (e.g. see line 5). When instantiating a device, we can override existing properties from the device (e.g. the name of the device as in line 6), and additional information regarding the implementation of the communication protocols. For instance, in line 9 we can see the specification of the Zigbee address for the plug instance. This set will contain all the devices existing in the EMS.

The `network` describes how the devices connect with each other. Usually, in EMS, we have a set of sensors connecting

```
1  using "devices/Plug:0.1"
2  using "devices/Hub:0.8"
3  system House1
4     devices:
5        tv as Plug:0.1 {
6           name = "TV"
7           location = "Living Room"
8           communication = {
9              zigTv as Zigbee = "DF:3B:00:11:22:33:FF:EE"
10       }}
11       fridge as Plug:0.1{ /*...*/ }
12       hub as Hub:0.8{
13          name = "Cloogy"
14          upload = every 48 unit=h
15          communication = {
16             zigHub as Zigbee = "CA:CC:00:44:02:77:FF:11"
17       }}
18    network:
19       tv -> hub via zigTV -> zigHub:
20          acquisitions:
21             InstCurrent every 10 unit=s
22             ...
23       fridge -> hub via zibFridge -> zigHub
24    communication:
25       mqtt: "house1.sensors.notifications/#"
26    properties:
27       name: "Cloogy House1"
28       description: "A house containing four devices"
```

Figure 3: Example of a system architecture design

to the managing hub, which is the one responsible to request measurements from the sensors. The network is represented as a directional graph that shows how the devices connect, and if the connection is one way of bidirectional. When measurements are to be acquired within a connection, the `acquisitions` attribute can be used, as shown in line 21 of Figure 3. Here, we have to define which measurement is to be acquired and the acquisition period.

The `communication` specifies how the DT manager obtains the data from the real system. This is the configuration for the `System Communication` tool. Currently, only the mqtt protocol is available, as exemplified in line 26.

Further properties can be specified using the `properties` attribute, where a name and a description can be defined, among other metadata.

The execution of the EMS compiler with a system definition - e.g. with the one defined in Figure 3 - will generate a set of assets that, when deployed, will perform as the DT of the target EMS. These assets include the necessary files to deploy a Ditto instance, the communication configuration to interact the real system with the DT, the database configuration and the Grafana configuration, including a set of pre-defined dashboards for quick data visualization.

5 DT Instantiation

A DT instance can be automatically deployed using the generated assets. This includes the files necessary to start the Ditto framework and start the communication with the real system. This first step is already enough to start managing and visualize the DT, as Ditto provides a GUI to see the current

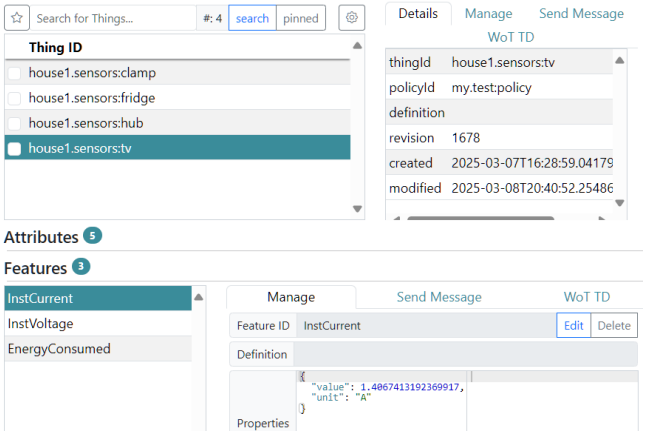


Figure 4: Management view of an EMS from the Ditto framework.

state of the devices - `Things` in Ditto. One important note is that the DSL actually abstracts many of the concepts required in a specification of a `Thing`. The code generation process guarantees that all necessary information is automatically generated.

Figure 4 shows an example of a deployed system in Ditto. The example is highlighting a plug sensor connected to a `tv`, where in the Features view it is possible to observe the last reading of the measurement `Instant Current` obtained from that device (see bottom-right side, `value` property).

Since ditto does not store the historical data of the obtained updates, a database is necessary. We use Clickhouse as the database as we intend to keep track of timeline series of measurements, and Grafana as the GUI for the timelines. The system is configured to insert a new instance of data in the database every time Ditto is updated. The database contains dedicated tables for the devices, which can be quickly queried to obtain information from a specific device, or a specific measurement type.

Figure 5 shows a dashboard in Grafana that was automatically generated by the EMS compiler. The dashboard shows three timelines, each one for a specific sensing device. Each timeline contains the values of measurements being acquired from that device. Taking advantage of Grafana features, can easily build new dashboards. For instance, a dashboard for a specific device that divides the measurements in separate graphics, and an inclusion of a statistical table. This means that both the GUI and the queries are customizable.

6 Conclusions

We presented an approach for modeling and managing DTs for EMS. The approach provides a DSL especially focused on EMS that provides an abstraction layer for EMS developers, simplifying their learning curve for building DTs. The approach is able to automatically provide assets and configurations for the necessary tools for management, communication, storage and visualization. The communication layer provides an interface between the real-life system (or simulation), while the database enables the system to keep track of the system history.

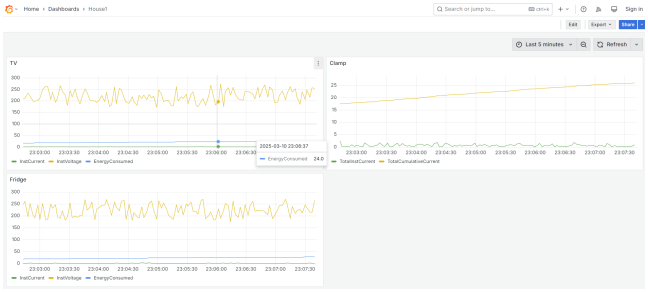


Figure 5: A timeline view of the measurements acquisition for an example EMS.

As future work, the communication layer process will be able to deal with more complex systems, using different protocols. The visualization layer will take into account other types of user interfaces, for instance, to have a high level view of the system overall and its communication. It is also planned to evolve the approach from a DS into a DT, allowing the communication of the DT with the real system. Furthermore, the approach will evolve to take into account more complex systems, especially in the context of smart communities (i.e. a group of houses and energy grids) and smartgrids.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments and feedback. This work was supported by the KDT Joint Undertaking (JU) program under grant agreement No 101140216 (project MATISSE), co-funded by the European Union's Horizon Europe research and innovation programme and the Portuguese Fundação para a Ciência e a Tecnologia (FCT).

References

[1] K. Chooruang and K. Meekul, "Design of an iot energy monitoring system," in *2018 16th Intl. Conf. on ICT and Knowledge Engineering*, pp. 1–4, 2018.

[2] D. Menniti, A. Pinnarelli, N. Sorrentino, P. Vizza, G. Barone, G. Brusco, S. Mendicino, L. Mendicino, and G. Polizzi, "Enabling technologies for energy communities: some experimental use cases," *Energies*, vol. 15, no. 17, p. 6374, 2022.

[3] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, "Digital twin modeling," *Journal of Manufacturing Systems*, vol. 64, pp. 372–389, 2022.

[4] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building digital twins of articulated objects from interaction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5616–5626, 2022.

[5] D. Raggett, "The web of things: Challenges and opportunities," *Computer*, vol. 48, no. 5, pp. 26–32, 2015.

[6] L. Pires Klein, A. Krivoglazova, L. Matos, J. Landeck, and M. de Azevedo, "A novel peer-to-peer energy sharing business model for the portuguese energy market," *Energies*, vol. 13, no. 1, 2020.

A Comparative Study of Reliable Predictive Models for Energy-Efficient MPC in Buildings

Alexandre Geraldo, António Casimiro, Alan Oliveira de Sá, José Cecílio, Pedro M. Ferreira

LASIGE, Faculty of Sciences, University of Lisbon, Portugal; email: ageraldo@lasige.di.fc.ul.pt, {casim, aodsa, jmcecilio, pmf}@ciencias.ulisboa.pt

Abstract

Unlike traditional controllers, Model Predictive Control (MPC) leverages predictive models to predict future system behavior, enabling proactive and efficient decision-making over a defined time horizon. Its effectiveness depends on the accuracy of the models used. Inaccurate predictions can compromise system reliability, reduce stability, and lead to suboptimal control decisions. This work considers the problem of optimizing energy efficiency while ensuring thermal comfort inside buildings, aiming at proposing a suitable MPC strategy in this context. This paper presents preliminary results towards this goal, comparing a gray-box Resistor-Capacitor (RC) model and a black-box Artificial Neural Network (ANN) trained on Pseudo-Random Binary Sequence (PRBS) generated data. Results show that while the RC model is interpretable, the ANN achieves superior long-term accuracy. Observing the Mean Absolute Error (MAE) curves and their standard deviation, we conclude that the reliable prediction horizon, which ensures a prediction error below $1^{\circ}C$*, is only 9.5 hours with the RC model, and increases to more than 24 hours with the ANN model, making it more suitable for MPC applications.*

Keywords: Model Predictive Control, Artificial Neural Networks, Resistor-Capacitor Model, Buildings.

1 Introduction

Buildings account for 40% of final energy consumption in the European Union and 36% of its energy-related greenhouse gas emissions, while 75% of buildings are still energy inefficient [1]. This highlights the urgent need to optimize energy efficiency in buildings. In addition, the increasing energy costs, environmental concerns, grid management challenges, and stricter regulations demand innovative solutions to improve energy efficiency.

Model Predictive Control (MPC) has gained significant attention as an advanced control strategy that optimizes energy consumption while ensuring occupant comfort. Unlike traditional controllers, MPC anticipates future system behavior using predictive models, allowing control decisions that optimize efficiency over a given prediction horizon. However, one of the main challenges in MPC implementation is its strong dependence on accurate models, which directly affects the control system's reliability, stability, and overall performance. An inaccurate model can lead to suboptimal energy usage, increased operational costs, and reduced thermal comfort.

Given the importance of reliable predictive models, a systematic methodology for constructing and evaluating regression models that accurately capture the building's thermal dynamics is essential. This study examines the performance of the most commonly used gray-box and black-box predictive models for thermal prediction in MPC. Specifically, we aim to assess their accuracy and effectiveness regarding prediction capabilities, particularly concerning reliable prediction horizons, contributing to more stable and energy-efficient building management. To achieve this, two distinct modeling approaches have been developed:

- A 2R2C thermal network model with parameters identified using Particle Swarm Optimization (PSO).
- An Artificial Neural Network (ANN)-based model.

2 Related Work

The performance of MPC depends on the accuracy and robustness of its predictive models. Error propagates through the prediction horizon, affecting stability and performance. The choice of modeling approach significantly impacts how well MPC handles uncertainties and disturbances.

White-box models are based on physical principles such as heat transfer and energy conservation, with parameters derived from building information such as geometry, material properties, and equipment specifications. Although they are theoretically sound, they are rarely used in MPC due to their high complexity and computational demands. For example, Ascione et al. [1] implemented an EnergyPlus-based white-box model for a nearly zero-energy building, but its computational cost made real-time control impractical, highlighting the need for simpler models.

Gray-box models combine simplified physical principles with parameter estimation from measured data. The Resistor-Capacitor (RC) network model is the most widely used gray-box approach for building thermal dynamics in MPC [2, 3, 4], offering a balance between accuracy and computational efficiency. These models are easier to develop due to established formulations but rely on accurate parameter identification. Various optimization techniques, such as Genetic

[1] https://eur-lex.europa.eu/eli/dir/2024/1275

Algorithms [2] and PSO [4], have been explored to improve parameter estimation.

Black-box models rely entirely on data-driven methods to learn building dynamics, making them more flexible but dependent on high-quality datasets. They also have been particularly effective in modeling complex, nonlinear behaviors such as occupancy and thermal comfort. Ferreira et al. [5] used Radial Basis Function Neural Networks to simplify PMV predictions and predict temperature and humidity, while Eini et al. [6] developed a network-based model for occupancy estimation. Arendt et al. [7] found that black-box models outperformed both white-box and gray-box approaches in several cases, demonstrating their practical advantages when sufficient data is available.

Dataset construction is crucial for both gray-box and black-box MPC modeling in buildings. Black-box models require diverse, advanced datasets, often using techniques like PRBS signals as demonstrated by Ferreira et al. [5] and Royer et al. [8], to capture complex behavior. In contrast, gray-box models use regular operation datasets that focus on basic parameters such as indoor and outdoor temperatures, energy use, and occupancy, as shown by Eini et al. [6] and Tang and Wang [2]. This shows that the modeling approach determines the type and quality of data needed.

This work builds on these findings by developing and directly comparing gray-box and black-box approaches, which are among the most practical and widely used models in MPC applications for building energy systems. The objective is to assess the reliability of these models over a prediction horizon to compare their performance and determine for how long they remain sufficiently reliable for MPC applications.

3 Methodology

The reliability of predictive models is crucial for effective MPC in building energy management. This section details the development and evaluation of two modeling approaches: a gray-box 2R2C thermal network model, optimized via PSO, and a black-box ANN model, both trained on a dataset generated using an EnergyPlus [2]-calibrated model of the library at the Faculty of Sciences, University of Lisbon. To generate this dataset, a PRBS signal generator was developed and applied in a simulation environment to excite the system, producing a highly dynamic dataset equivalent to one year of operational data. The goal is to develop methodologies to build these models and assess their predictive reliability for MPC applications while leveraging a simulation environment to avoid downtime in the real building. The following subsections outline their formulation, training, and evaluation.

3.1 RC Model

The RC model is a widely used gray-box method for modeling building thermal dynamics, using thermal resistance (R) and capacitance (C) to estimate heat transfer and energy storage.

In this study, a **2R2C thermal network model** was selected due to its effectiveness in representing the thermal behavior of buildings. The schematic representation of the model is shown in Figure 1.

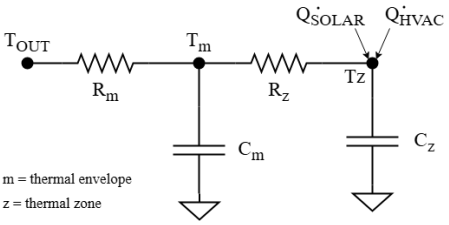


Figure 1: Electrical schematic representation of the 2R2C thermal network model.

The system's differential equations were discretized using the Forward Euler method, leading to the following discrete-time representation:

$$T_m^{k+1} = T_m^k + \frac{\Delta t}{C_1}\left(\frac{T_{\text{out}}^k - T_m^k}{R_1} - \frac{T_m^k - T_z^k}{R_2}\right),$$
$$T_z^{k+1} = T_z^k + \frac{\Delta t}{C_2}\left(\frac{T_m^k - T_z^k}{R_2} + Q_{\text{HVAC}}^k + F_{\text{solar}} Q_{\text{solar}}^k\right) \quad (1)$$

The term Q_{HVAC} represents the heating or cooling power supplied by the HVAC system. The solar heat gain Q_{solar} accounts for the effect of solar radiation on the building.

3.1.1 Parameter Identification

To find the parameters of the 2R2C thermal network model, a PSO algorithm was employed using the PySwarms library [9], a Python-based toolkit for PSO-based optimization. The optimization minimized the Mean Squared Error (MSE) between the simulated and measured indoor temperatures by tuning the model parameters, thermal resistances (R_m, R_z), thermal capacitances (C_m, C_z), and the solar gain factor (F_{solar}). PSO optimization used 256 particles for 100 iterations, with an exponentially decaying inertia weight and adaptive c_1 and c_2 coefficients to balance exploration and exploitation.

Preliminary runs were conducted to refine search space boundaries, particularly for the thermal capacitances (C_m, C_z). The optimizer initially selected excessively high values, leading to an overly dampened response and unrealistic or no temperature variations. To mitigate this, the upper bounds of C_m and C_z were restricted, ensuring physically meaningful solutions. After finalizing these constraints, multiple optimization runs were performed, and the best-fitting parameters were selected.

3.2 Artificial Neural Network Model

ANNs are commonly used black-box models for building thermal dynamics and learning complex relationships from data without relying on physical insights. In this study, a fully connected feedforward ANN was developed using `PyTorch` [3].

The input layer consists of 322 time-lagged input variable values, which include both current and past values (up to 24 hours) of T_z (indoor temperature), T_{out} (outdoor temperature), T_{HVAC} (HVAC setpoint temperature), and Q_{solar} (direct solar radiation). An inverse-sampling strategy was employed to

[2] https://energyplus.net/
[3] https://pytorch.org/

prioritize more recent values over those further back in the 24-hour period. The model has two hidden layers, each using the ReLU activation function. The output layer is designed to predict the next-step indoor temperature T_z^{k+1}.

The model was trained using one year of data, which was split into 80% for training and 20% for testing. This dataset is highly dynamic, as it was generated with a PRBS signal, ensuring various conditions. To maintain seasonal representation in both sets, the data was shuffled deterministically. MSE was chosen for the loss function since it is a standard approach for regression tasks. Finally, optimization was conducted with the *Adam* optimizer, employing the default learning rate and applying weight decay proportional to the learning rate divided by the number of epochs.

3.2.1 Training

The training followed a structured methodology consisting of systematically optimizing the network architecture and batch size while minimizing training time and ensuring generalization. The training scripts used in the development of the ANN were built leveraging the `Optuna` library [10] in a multiprocess distributed environment, enabling efficient GPU utilization and faster parallel training.

A grid search approach was used to identify the size of the two hidden layers (initially, both layers had the same number of neurons). Once a suitable range of hidden layer sizes was identified, different batch sizes were evaluated. Lastly, a refined grid search was performed, where each hidden layer was varied independently within the best-performing size range identified in the first stage. This step was conducted using the optimal batch size from the previous stage.

Notably, at this stage, the grid search did not include other hyperparameters, such as the learning rate. Initial results suggested that further adjustments would have a negligible impact, as the loss values were already well optimized at this stage.

3.3 Model Validation

To evaluate the long-term predictive reliability of the models, the performance was assessed over a 24-hour prediction horizon spanning 6 months. This was accomplished using a sliding window approach. At each time step during the 6 months, the models generated predictions for the following 24 hours.

Model performance was quantified using the Mean Absolute Error (MAE) and its standard deviation to analyze prediction variability. Maximum and minimum absolute error curves were computed to establish error bounds. To evaluate robustness over the full horizon, the integral of the MAE curve was used as a metric, where a lower integral indicates greater long-term reliability.

The results are visualized in error evolution plots, highlighting how MAE changes over the prediction and depicting prediction uncertainty through ± 1 standard deviation shading. This methodology allows for a direct comparison of the models, providing insight into their ability to maintain accuracy over extended forecasts.

4 Results

4.1 Identified Parameters

For the ANN model, the grid search showed that hidden layers up to 50 neurons achieved the best performance, while larger networks offered significantly worse results. The optimal batch size was 256 to 512, balancing accuracy and training efficiency. Larger batch sizes provided no speedup and slightly worsened performance, while smaller batch sizes significantly degraded both accuracy and training time.

After multiple runs, the optimal RC model parameters were identified through PSO, refining the search space bounds to prevent overestimation of capacitance. The final values, optimized for minimum MSE, are presented in Table 1.

Table 1: Best RC Model Parameters

R_m	R_z	C_m	C_z	F_{solar}	MSE
4.139	0.658	120963.83	40427.24	0.0519	0.142

4.2 Model Evaluation

The comparison between the RC and the ANN model reveals that the ANN model is more robust over the 24-hour prediction horizon. The MAE curve for RC (Figure 2) is consistently higher and has a steeper slope than ANN (Figure 3), indicating faster accuracy degradation. The first-horizon predictions (Figures 4 and 5) further illustrate that the ANN model follows the reference temperature more closely.

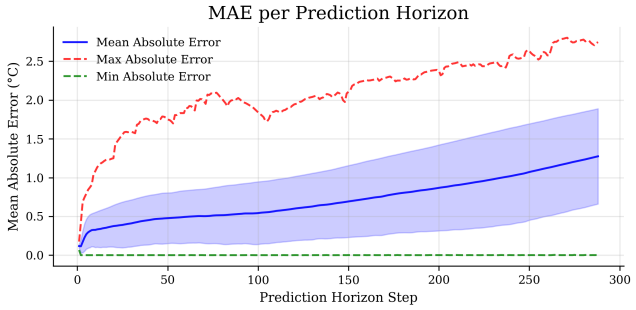


Figure 2: RC Model: MAE per Prediction Horizon

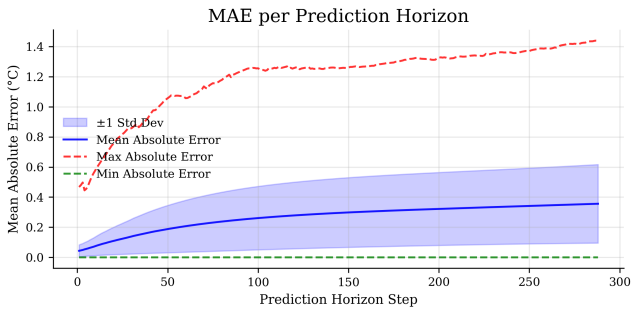


Figure 3: NN Model: MAE per Prediction Horizon

A quantitative evaluation using the integral of the MAE curves confirms this trend: the RC model reaches 207, while the NN model achieves 77, indicating a lower cumulative error over time.

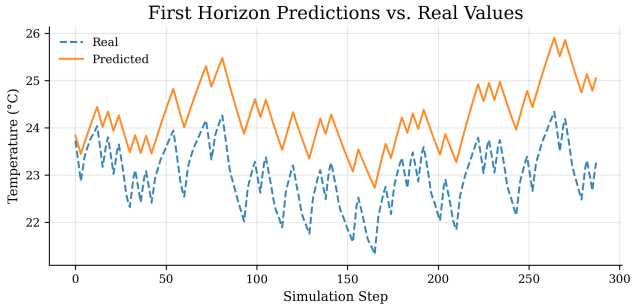


Figure 4: RC Model: Horizon Predictions vs. Real Values

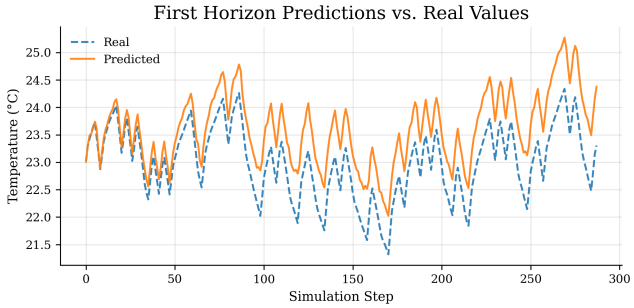


Figure 5: NN Model: Horizon Predictions vs. Real Values

A $1.0°C$ error threshold was defined based on typical $\pm 0.5°C$ HVAC sensor accuracy to assess practical reliability for MPC. Observing the MAE curve and its standard deviation, the RC model surpasses this limit after 115 steps (9.5 hours). In contrast, the ANN model remains below $0.6°C$ even at 24 hours, confirming its long-term predictive reliability. The RC model remains viable for most MPC applications, where typical prediction horizons range between 3 to 5 hours.

5 Conclusion and Future Work

This work aims to enhance the reliability of predictive models for Model Predictive Control, ultimately contributing to more stable and energy-efficient building management. We compared the prediction horizon capabilities of RC and ANN modeling approaches for building thermal dynamics in the context of MPC. While the RC model provides a simple and interpretable representation, the ANN model demonstrated greater predictive accuracy, consistently outperforming the RC model. These results indicate that the ANN is more suitable for long-term predictions in MPC applications.

However, this analysis was conducted under a best-case scenario, considering only model prediction errors for indoor temperature. In real-world applications, additional uncertainties, such as outdoor weather forecasts and internal load variations, introduce further error sources. Future work will incorporate multi-variable predictions to better assess model reliability over extended prediction horizons, accounting for these additional uncertainties.

Aiming at further improvement of predictive performance, future work will also explore Physics-Informed Neural Networks (PINNs), which integrate physical insights into the learning process to enhance generalization and robustness, particularly in low-data scenarios. Additionally, training methodologies will be refined by embedding reliability validation directly into the model selection process, ensuring that chosen models maintain accuracy across extended horizons.

Acknowledgment

This work was supported by FCT through the LASIGE Research Unit, ref. UID/408/20253.

References

[1] F. Ascione, R. F. De Masi, V. Festa, G. M. Mauro, and G. P. Vanoli, "Optimizing space cooling of a nearly zero energy building via model predictive control: Energy cost *vs* comfort," *Energy and Buildings*, vol. 278, p. 112664, Jan. 2023.

[2] R. Tang and S. Wang, "Model predictive control for thermal energy storage and thermal comfort optimization of building demand response in smart grids," *Applied Energy*, vol. 242, pp. 873–882, May 2019.

[3] S. Abobakr, M. Alosta, M. A. Abdelkefi, A. E. Kaouachi, and L. Sboui, "Mpc-based efficient energy control and cost estimation of hvac in buildings," in *2023 IEEE Third International Conference on Signal, Control and Communication (SCC)*, pp. 1–5, 2023.

[4] K. Micheneau and A. Boodi, "Optimizing comfort and energy efficiency: The impact of model accuracy on multi-objective mpc," in *2024 6th Global Power, Energy and Communication Conference (GPECOM)*, pp. 659–664, 2024.

[5] P. Ferreira, A. Ruano, S. Silva, and E. Conceição, "Neural networks based predictive control for thermal comfort and energy savings in public buildings," *Energy and Buildings*, vol. 55, pp. 238–251, 2012. Cool Roofs, Cool Pavements, Cool Cities, and Cool World.

[6] R. Eini and S. Abdelwahed, "Learning-based model predictive control for smart building thermal management," in *2019 IEEE 16th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT and AI (HONET-ICT)*, pp. 038–042, 2019.

[7] K. Arendt, M. Jradi, H. Shaker, and C. Veje, "Comparative Analysis of White-, Gray- and Black-Box Models for Thermal Simulation of Indoor Environment: Teaching Building Case Study," vol. 8 of *SimBuild Conference*, pp. 173–180, ASHRAE/IBPSA-USA, 2018.

[8] S. Royer, S. Thil, T. Talbert, and M. Polit, "A procedure for modeling buildings and their thermal zones using co-simulation and system identification," *Energy and Buildings*, vol. 78, pp. 231–237, Aug. 2014.

[9] L. J. V. Miranda, "PySwarms, a research-toolkit for Particle Swarm Optimization in Python," *Journal of Open Source Software*, vol. 3, 2018.

[10] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

Dynamic System Model Generation for Online Fault Detection and Diagnosis of Robotic Systems

Johannes Kohl, Georg Muck, Georg Jäger, Sebastian Zug

Freiberg University of Mining & Technology, Institute for Computer Science, Akademiestraße 6, 09599 Freiberg, Germany; email: {johannes-georg.kohl, georg.muck, georg.jaeger, sebastian.zug}@informatik.tu-freiberg.de

Abstract

With the rapid development of more complex robots, Fault Detection and Diagnosis becomes increasingly harder. Especially the need for predetermined models and historic data is problematic because they do not encompass the dynamic and fast-changing nature of such systems. To this end, we propose a concept that actively generates a dynamic system model at runtime and utilizes it to locate root causes. The goal is to be applicable to all kinds of robotic systems that share a similar software design. Additionally, it should exhibit minimal overhead and enhance independence from expert attention.

1 Motivation

Mobile robotic systems are increasingly deployed in open environments, requiring them to handle a variety of external influences.This complexity increases the potential for diverse faults [1], ranging from hardware and actuator failures to software and interaction-related issues. Detecting these faults at design time is challenging due to the complexity of such systems. However, undetected faults threaten the reliability and safety of robotic systems, making online Fault Detection and Diagnosis (FDD) essential [2].

Conventional FDD, however, struggles with the complexity of mobile robots and the dynamic nature of rapid prototyping. This is due to the characteristics of continuous changes from this approach. To address these challenges, this article describes a work-in-progress approach to dynamic FDD for mobile robots, using a Digital Shadow – an aggregated, passively generated data footprint that reflects the robotic system [3]. We aim at an FDD design that minimizes computational overhead as well as expert knowledge and is applicable to a variety of robots. For that reason, Section 3.1 introduces a generalized mathematical meta-model to encompass a broad range of Distributed Systems (DSyss) like ROS2.

In the following, we analyse *System Modelling* (SM) for robotics in Section 2.1. Next, we review previous developments in FDD and identify the necessity of an online system model in Section 2.2. Based on these insights, we construct a mathematical meta-model that shadows a generic DSys in Section 3.1, apply it to a software layer in Section 3.2 and suggest how it can be utilized effectively in Section 3.3. We conclude the article with an outlook on planned future work in Section 4.

This work was partially funded by the German Federal Ministry for Digital and Transport within the mFUND program (grant no. 19FS2025A - Project Ready for Smart City Robots).

2 State of the Art on SM and FDD

To review the current state of FDD and SM, we focus on Model Based System Engineering (MBSE) and Root Cause Analysis (RCA). This enables us to identify techniques aiming for minimal computational overhead and expert knowledge. Moreover, this allows the identification of requirements by assessing the shortcomings of the proposed methods.

Within this work, we use the following terminology: A *robot* is the physical platform hosting the *Robotics Framework* (RF), which describes a software infrastructure layer that handles communication, lifetime, etc. like ROS2 [4] or ORCA [5]. A *Distributed System* (DSys) in turn describes the software stack built for the robot and is running in the RF. A DSys consists of *Components* which perform system tasks and generate process data, and *Distributors* which facilitate the communication between components, e.g. Topics in ROS2 [4]. Furthermore, we follow the definitions of [6]: A *symptom* is an indicator of a failure, like service degradation or alarms. A *failure* is a deviation between actual and expected system behaviour. An *error* is the part of the system state that may cause a failure. A *fault* is a possible or real cause of an error.

2.1 System Modelling

Building on this foundation, we explore approaches of MBSE in robotics to develop a dynamic, data-driven model that reduces expert knowledge and enables the creation of a Digital Shadow. As the goal is to systematically detect and diagnose faults, MBSE provides a structured framework to formally model system behaviour, interactions and failure propagation [7]. It is an approach that enables the establishment of models for various use cases, ensuring their structure and maintainability even in complex scenarios [7] [8]. This means MBSE supports the identification of faults at design time using a holistic view [9]. Existing work widely adopts *Systems Modelling Language* as the modelling language for system representation due to its expressiveness and versatility [8].

Previous studies have largely focused on the interaction between hardware and software, e.g. [10], [11] and [12]. [13]

highlights the role of Build-In Tests (BIT), which are embedded within the system architecture, predominantly during its design phase, to detect and diagnose failures in real time. Thus, they maximize the number of detectable faults and enable more efficient maintenance by reducing diagnostic latency. The author also notes the challenge of retrofitting BIT capabilities into already existing systems. Since the targeted systems have already passed their design phase, the approach is not ideal.

An alternative is to use a Digital Shadow, which is a virtual representation that mirrors the physical system at runtime. Contrary to a Digital Twin [3], the automated data flow only reaches from the physical system to the virtual model and therefore is not bidirectional. In conclusion, the Digital Shadow cannot manipulate the physical system as a Digital Twin could. Both use simulations for the prediction of faults or maintenance needs (predictive maintenance).

[14] uses MBSE to construct a scalable Digital Twin of industrial robots that is used to gather failure data and enable predictive maintenance. Furthermore, the authors propose a modular, hierarchical and generic approach to enhance reusability, scalability, and interoperability across different industrial robot applications and lifecycle stages. However, the approach describes the creation of a Digital Twin throughout the entire lifecycle, whereas our approach targets systems beyond the design phase.

In summary, despite MBSE's adoption in robotics, the complexity of modelling robotic systems leads to the lack of such a model [11]. To fill the gap, we propose the use of MBSE

to create an abstract meta-model in Section 3.1. It aims to enable the automated generation of system models at runtime and provide knowledge that can be used to detect and localize faults.

2.2 Root Cause Analysis

FDD is a generic term for inductive methods, postulating a fault and analysing its effect on the system, and deductive methods, postulating a failure and analysing which errors or faults may influence it [15]. *Failure Mode and Effects Analysis* is one such inductive analysis method focusing on the system's risks. It is usually conducted in the design phase and thus not well suited for automation. *Fault Tree Analysis* and *Event Tree Analysis* are deductive and inductive respectively [15] and analyse the system's causal connections. They can be used at runtime to identify a *Root Cause* (RC), a fault that causes another but is not caused by one [16]. Knowledge about RCs can be used to enable subsequent components to initiate appropriate countermeasures.

Such methods that aim to identify RCs from a set of symptoms are categorized as RCA [16]. RCA generally relies on a *System Object Taxonomy*, i.e. a graph modelling the targeted DSys using objects, representing its components and distributors, as well as their interdependencies. This is needed to retrieve a *Fault Trajectory* (FT), i.e. an ordered set of system objects exhibiting symptoms, starting at the initially detected symptom and ending in a RC candidate. In order to detect these symptoms, many RCA algorithms rely on signals carrying information about the symptom, called *Alerts*. While no RCA algorithms for robotics were found in the literature review, related proposals exist in cloud computing and microservices [17] [18] [19]. These algorithms are promising, as their approach of decomposing monolithic structures into modular services with lightweight communication mirrors the distributed architecture of RFs.

OpenRCA [17] identifies RCs by correlating symptoms in time series of alerts, leveraging multiple methods to balance their strengths and weaknesses. It maps symptoms to a system topology for tracing FTs.

[18] proposes GRANO (**Gra**ph **Ano**maly) which aims to score each system topology component with a *Root Cause Relevance*, i.e. a weight to get taken into account when reviewing FTs. It is calculated based on the amount and severity of alerts emitted for each component. While this allows an efficient search for RCs it still performed manually.

Another approach is taken by MicroRCA [19] which only looks at communication timing and service performance data. These metrics are analysed for symptoms like anomalous CPU utilization to be annotated into a system topology graph. This annotated graph in turn is analysed by the Personalized PageRank [20] algorithm, sorting inputs based on their importance in order to find RCs.

Even though the above approaches show promising evaluation results, their strategies have two main deficiencies in regard to the stated requirements of this article. Firstly, they expect the RF to supply facilities that emit alerts on detected symptoms. Since this cannot be done without expert knowledge, we propose the use of a plugin system to detect faults. Secondly, OpenRCA and GRANO require a holistic system object taxonomy, assumed to be statically provided by expert knowledge.

To align with our autonomy goals and to reduce expert knowledge, a meta-model to dynamically build and maintain a graph of the distributed system is defined in the next section.

3 Concept

For clarity, our approach is thematically subdivided into the generation of the system model (Section 3.2) and failure analysis (Section 3.3). The former focuses on modelling the system to support FDD. The latter aims to utilize the created model to extract the FT. The underlying conceptual meta-model is described in Section 3.1.

3.1 System Model

To ensure a structured yet adaptable representation of a DSys, we propose a meta-model that abstracts system components.

In this work, we model a DSys as a directed graph $G = (M, E, \theta)$. The graph consists of a set of members $M = M_A \cup M_P$, where $M_A \cap M_P = \emptyset$, and a set of not disjoint edges $E = E_0 \cup E_i$ with $0 < i < N \in \mathbb{N}_0$. The attribute function is defined as $\theta = \theta_A \cup \theta_P$, with $\theta_A \colon M_A \to A_A \subseteq \mathbb{R}^a$ and $\theta_P \colon M_P \to A_P \subseteq \mathbb{R}^p$, where $a, p \in \mathbb{N}$.

The system differentiates between two types of members. Active members execute or control operations, e.g. Nodes in ROS2 [4], while passive members enable data distribution or serve as organisational units, e.g. Topics in ROS2 [4].

The graph contains multiple types of communication edges, defined as $E_0 = E_{send} \cup E_{pub} \cup E_{sub}$. Specifically, sending edges are given by $E_{send} = (M_A \times M_A) \times \{0\}$, publishing edges by $E_{pub} = (M_A \times M_P) \times \{0\}$, and subscribing edges by $E_{sub} = (M_P \times M_A) \times \{0\}$. Each edge represents a one-to-one connection, but since passive members act as intermediaries, they enable many-to-many connections between active members M_A. The edge-induced subgraph $G_0 = (M_0, E_0, \theta)$ represents the communication structure of the distributed system, where $\forall m, n \in M_0 \subseteq M : (m, n) \in E_0 \wedge (n, m) \in E_0$. Additional edges model other relationships and are defined as $E_i = (M \times M) \times i$. The corresponding edge-induced subgraph $T_i = (M_i, E_i, \theta)$ of G forms a rooted tree, where the edges are given by $E_i = \{(v, u) | ((v, u), i) \in E\}$, ensuring that T_i contains only the vertices M_i, with M_i formed analogously to M_0 above.

3.2 Data Aggregation

Building on the established meta-model, we now focus on modelling conceptual data aggregation. Using a top-down approach, we enable hierarchical observation, allowing broad system analysis that becomes more precise as needed.

3.2.1 Top-Down Aggregation for FDD

The top-down approach is characterized by its initiation at the abstract levels, progressing in an incremental manner towards more detailed levels. The approach can be applied by using the edge-induced trees described in Section 3.1. Let $C_i(v) = \{u \in M_i | (v, u) \in E_i\}$ denote the children of $v \in M_i$. The accumulated property function $\psi_i(v) \colon M_i \to A$ is then recursively defined as $\psi_i(v) = \sum_{c \in C_i(v)} (\psi_i(c) + \theta(c))$. This recursive formulation closely resembles a depth-first search.

Building on the meta-model (Section 3.1) and the top-down approach described above, we now focus on projecting the conceptual model onto the software layer.

3.2.2 Projection to the Software Layer

In this approach, the distributed system described in Section 2 runs on an Operating System (OS) and consists of *components* and *distributors*. The OS is expected to provide the necessary infrastructure, resources, and services for execution. In this work, we focus on the Unix family of OSs.

A component corresponds to an active Member $m_{0,A} \in M_{0,A} = M_0 \cap M_A$ in the edge-induced subgraph G_0 on a software layer. It has a distinct name and can be associated with a process on the OS. Passive Members $m_{0,P} \in M_{0,P} = M_0 \cap M_P$ in G_0 serve as message distributors and are termed distributors. Communication occurs via the edges E_0.

After projecting the meta-model onto the software layer, the next sections will examine various aggregation techniques.

3.2.3 Fundamental Aggregation

The fundamental approach involves establishing relationships between components and distributors, mapping the directed connections between individual components including associated distributors. These connections enable querying the Graph for all components $r \in G_0$ predecessors $M_0 \ni r_{pre} = \{m_0 \in M_0 | \exists E_0 \ni e_0 = (m_0, r)\}$, active predecessors $M_{0,A} \ni r_{pre} = \{m_{0,A} \in M_{0,A} | \exists E_0 \ni e_0 = (m_{0,A}, r)\}$ and passive predecessors $M_{0,P} \ni r_{pre} = \{m_{0,P} \in M_{0,P} | \exists E_0 \ni e_0 = (m_{0,P}, r)\}$ and successors in an analogue way. This enables subsequent FDDs to derive a submodel containing the necessary information.

Another perspective will be introduced next by using a tree T_i.

3.2.4 Process Tree as a top-down approach

In a Unix-based OS, all processes except the root process are linked to a parent, forming a process tree. Since related components and their processes are typically grouped during their startup (e.g., via initialization scripts), they share common ancestors. This tree structure can be created by tracing the parent processes of M_A. If multiple root processes exist, a common virtual root can be introduced.

The provided structures described in Section 3.2.3 and Section 3.2.4 are used to detect faults in the following section.

3.3 Fault Detection and Root Cause Analysis

In this section, we present a concept to address the problems identified in Section 2.2. It leverages a dynamic subgraph of the system model and proposes a symptom detection method to generate alerts, ensuring independence from the RFs's provision of such a facility.

In the following, we define the subgraph $G_j = (M_j, E_j)$ with members $M_j \subseteq M$, edges $E_j \subseteq E : \forall((m_0, m_1), 0) \in E_j \colon m_0, m_1 \in M_j$ and iteration $j \in \mathbb{N}_0$. Here, an iteration describes a stage of the subgraph's construction.

Next, the in Section 3.3.1 explained symptom detection is part of the dynamic allocation of the subgraph in Section 3.3.2, which in turn is used to extract FTs in Section 3.3.3.

3.3.1 Symptom Detection

As discussed in Section 2.2, providing a comprehensive symptom detection is beyond the scope of this article, making reliance on expert knowledge unavoidable. To use this knowledge efficiently, we propose a plugin system to be populated by the expert, which will emit alerts for detected symptoms.

For the purpose of providing information to the symptom detection, we define the watchlist $W_j \subseteq M$ which provides the attributes $\theta(w \in W_j)$ to the plugins. Alerts emitted by the symptom detection get stored in the alert database saving their origin and a timestamp, in order to extract and analyze time series of alerts per member later.

3.3.2 Dynamic Subgraph Allocation

The alerts emitted by the symptom detection also are used to expand the subgraph. This means that when an alert gets emitted for member $m_{alert} \in W_j$, the sets for the next iteration will be defined as follows: A) $M_{i+1} = M_j \cup \{m_{alert}\}$, B) $E_{i+1} = E_j \cup \{((m, m_{alert}), 0) \in E | m \in M_j\}$ and C) $W_{i+1} = W_j \cup \{m \in M | ((m_{alert}, m), 0) \in E\}$.

Iteration $i = 0$ reflects a special case: since the watchlist is empty, nothing can be added to the subgraph. Thus, an initialization is needed, which can be done in two ways. Firstly, by directly populating the watchlist with members from a configuration based on factors like importance or failure probability. Secondly, by identifying components in the process tree, extracted from G in Section 3.2.4, with anomalous metrics using process monitoring.

3.3.3 Fault Trajectory Extraction

At iteration $j_{extract}$, the subgraph $G_{j_{extract}}$ can be used to trace a FT and by extension a RC. An assortment of different symptom correlation methods can be used in conjunction to trace FTs. One such method is Symptom Co-Occurrence which aims to detect if two sets of time series of alerts co-occurred, using an Iterative Closest Points algorithm adapted to one-dimensional space. Another method is Symptom Time Lag Analysis that aims to detect symptom pairs by comparing their time lag to a time lag distribution. The retrieved FTs are being ranked by the average strength of their dependencies and length [17].

4 Conclusion and Future Work

Within this work we defined a mathematical meta-model that aims to universally represent Distributed Systems in order to facilitate dynamic Fault Detection and Diagnosis in the context of rapid prototyping.

We endeavour to expand our present model. This enhancement will facilitate a more comprehensive analysis of network activity. Additionally, we will investigate methods to ensure secure and efficient data aggregation and synchronization from various clients, addressing potential challenges related to data synchronization.

Furthermore, the RCA algorithm needs to be finalized and refined. This especially necessitates more research about alternative symptom correlation mechanisms and possible automations of the initialization. This also implies testing the feasibility of correlation methods established in Section 2.2 for online usage or the addition of an offline component to aid the live execution. Another goal would be adding functionality to differentiate between permanent and non-permanent faults and handle them separately.

The subsequent steps will be to focus on the aforementioned points and to implement the pipeline in ROS2. Based on the implementation the pipeline will be tested in multiple real-world scenarios and simulations to evaluate its applicability on a broad range of robotic systems.

References

[1] E. Khalastchi and M. Kalech, "On fault detection and diagnosis in robotic systems," *ACM Comput. Surv.*, vol. 51, Jan. 2018.

[2] R. Golombek, S. Wrede, M. Hanheide, and M. Heckmann, "Online data-driven fault detection for robotic systems," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3011–3016, 2011.

[3] V. Lopez and A. Akundi, "A conceptual model-based systems engineering (mbse) approach to develop digital twins," in *2022 IEEE International Systems Conference (SysCon)*, pp. 1–5, 2022.

[4] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, May 2022.

[5] A. Makarenko, A. Brooks, and T. Kaupp, "Orca: Components for robotics," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 163–168, Citeseer, 2006.

[6] A. Avizienis, J.-C. Laprie, B. Randell, *et al.*, "Fundamental concepts of dependability," *Technical Report Series-University of Newcastle upon Tyne Computing Science*, 2001.

[7] K. Henderson and A. Salado, "Value and benefits of model-based systems engineering (mbse): Evidence from the literature," *Systems Engineering*, vol. 24, no. 1, pp. 51–66, 2021.

[8] A. L. Ramos, J. V. Ferreira, and J. Barceló, "Model-based systems engineering: An emerging approach for modern systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 101–111, 2011.

[9] A. H. de Andrade Melani and G. F. M. de Souza, "Obtaining fault trees through sysml diagrams: A mbse approach for reliability analysis," in *2020 Annual Reliability and Maintainability Symposium (RAMS)*, pp. 1–5, IEEE, 2020.

[10] L. Brisacier-Porchon and O. Hammami, "Tackling optimization and system-driven engineering in coupling physical constraints with mbse: The case of a mobile autonomous line of products," in *Conference on Systems Engineering Research*, pp. 441–459, Springer, 2023.

[11] K. Aloui, M. Hammadi, A. Guizani, T. Soriano, and M. Haddar, "Development of an agv system using mbse method and multi-agents' technology," in *International Conference Design and Modeling of Mechanical Systems*, pp. 103–114, Springer, 2021.

[12] P. K. M. Sekar and J. S. Baras, "Model-based systems engineering simulation framework for robot grasping," in *INCOSE International Symposium*, vol. 32, pp. 82–89, Wiley Online Library, 2022.

[13] N. Mahmood, S. Cimtalay, and D. N. Mavris, "Model based systems engineering (mbse) applied to fault detection analysis of vehicle subsystems," in *2022 IEEE Aerospace Conference (AERO)*, pp. 1–11, IEEE, 2022.

[14] X. Zhang, B. Wu, X. Zhang, J. Duan, C. Wan, and Y. Hu, "An effective mbse approach for constructing industrial robot digital twin system," *Robotics and Computer-Integrated Manufacturing*, vol. 80, p. 102455, 2023.

[15] L. Xing and S. V. Amari, *Fault Tree Analysis*, pp. 595–620. London: Springer London, 2008.

[16] M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada, "Survey on models and techniques for root-cause analysis," *CoRR*, vol. abs/1701.08546, 2017.

[17] B. Żurkowski and K. Zieliński, "Root cause analysis for cloud-native applications," *IEEE Transactions on Cloud Computing*, vol. 12, no. 1, pp. 232–250, 2024.

[18] H. Wang, P. Nguyen, J. Li, S. Kopru, G. Zhang, S. Katariya, and S. Ben-Romdhane, "Grano: interactive graph-based root cause analysis for cloud-native distributed data platform," *Proc. VLDB Endow.*, vol. 12, pp. 1942–1945, Aug. 2019.

[19] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "Microrca: Root cause localization of performance issues in microservices," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, 2020.

[20] G. Jeh and J. Widom, "Scaling personalized web search," in *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, (New York, NY, USA), p. 271–279, Association for Computing Machinery, 2003.

Verified SAT Redundancy Checking in SPARK*

Johannes K. Fichte
Artificial Intelligence and Integrated Computing Systems (AIICS), Department of Computer Science (IDA)
Linköping University, Sweden; email: johannes.fichte@liu.se

Tobias Philipp
SINA Development and Verification Team, Division Defence & Space
secunet Security Networks AG, Germany; email: tobias.philipp@secunet.com

Abstract

Boolean satisfiability (SAT) asks to decide whether a formula can be evaluated to true. In certified SAT solving, one additionally asks to provide a certificate or verifiable proof to ensure correctness of the solver's output. A crucial component in this process are fast verified, proof checkers. However, constructing efficient, formally verified checkers remains challenging.

In this paper, we study whether the Ada/SPARK programming language can enable reliable, efficient, verifiable proof checkers. SPARK 2014, with its design-by-contract capabilities, allows to express core concepts of propositional logic and modern redundancy criteria with functional correctness. We introduce a methodology for implementing SAT specifications in SPARK, accompanied by formal proofs with a focus on Resolution Asymmetric Tautologies (RAT). While a complete verified proof checker is not yet realized, our work establishes the foundational framework necessary for its future development, and demonstrates that strong correctness guarantees can be realized in SPARK.

1 Introduction

The *Boolean satisfiability (*SAT*)* problem is a fundamental problem in computer science, discrete mathematics, and artificial intelligence. SAT asks, given a propositional formula, whether there is a way to assign the variables to true or false such that the formula evaluates to true.

Example 1 *Consider the formula $F = (p \vee q) \wedge (\neg p \vee \neg q)$. If we map p to true and q to false, F evaluates to true making it a satisfying assignment resulting in F being satisfiable.*

SAT is the first problem shown to be NP-*complete* [1, 2] making it the canonical problem for all problems in NP and central to complexity theory. Practical SAT solving has significantly advanced over the last decades [3, 4] fostered by an annual competition[1], with leading solvers such as Kissat [5] and CaDiCaL [6]. Solver performance has improved to the extent of being widespread tools in industry, e.g., hardware and software verification [7], planning [8], bioinformatics [9], cryptoanalysis [10]. Today's solvers consist of a highly tuned systematic search procedure augmented with *conflict driven clause learning (CDCL)* [11], *clause removal* [12, 13, 14], *formula simplification techniques* [15], and many more very elaborate techniques that are non-trivial conceptually and extremely complex on the implementation level. Thus, bugs can be easily introduced. If the formula is satisfiable and the solver finds a satisfying assignment, we can efficiently check whether the solver's decision was correct. But if the solver decides that the given formula is unsatisfiable, we cannot simply ask the solver to output all 2^n possible truth assignments for an input formula on n variables. Therefore, several (practical) proof systems have been introduced over the last couple of years [16, 17, 18, 19, 20]. Commonly, a proof is a sequence of mathematically valid formulas starting from the given premises and ending with the expected statement. Clearly, we aim for only constructing correct proofs in such a way that we can easily verify that the given sequence is valid. For a propositional formulas, this is a sequence of formulas starting with the input formula and ending with a formula that is trivially unsatisfiable. In practice, the input formula is not output again into a certificate and one only records the additional steps of a solver. Since unsatisfiability proofs have been added to modern SAT solvers, the reliability increased significantly allowing that unsatisfiability results can independently be checked by proof checkers. In turn, verified proof checkers contribute significantly to robustness in reliable applications, offering trusted explanations and supporting the integrity of software systems.

In Table 1, we provide an overview on recent practical techniques for unsatisfiability proofs of SAT solvers, which evolved notably over time. Initial approaches and their proof formats, such as DRUP [21] and DRAT [15, 22], have been central to practical solving, but face challenges associated with efficiently checking and formally verifying proof checkers [23]. Advanced methods aim to address these challenges by reducing the search space (for relevant clauses and linear resolution derivations). Such formats are Resolution Asymmetric Tautology (RAT) [15], GRIT [24], LRAT [25], and FRAT [26]. The LRAT format can be efficiently generated by SAT solvers [27] and checked by verified proof checkers, which are implemented or verified using theorem provers such as ACL2 and Coq [28], as well as and CakeML [29].

*Our implementation is publicly available on Software Heritage and Github: `https://archive.softwareheritage.org/browse/origin/https://github.com/tobiasphilipp/lrat-spark`

[1] `https://satcompetition.github.io/`

But the most efficient verified checker, *cakelpr* [30], still performs roughly threefold slower than unverified C implementations [30]. Thus, we are left with two types of proof checkers (a) a verified but much slower one and (b) an unverified but significantly faster one.

In this paper, we address this gap and consider whether a verified proof checker in Ada/SPARK is feasible. Ada is an imperative and object-oriented programming language that supports design by contract [33]. This allows users to formally specify how procedures, functions, and data types should behave. We express this behavior through pre- or postconditions, invariants, parameter modes, and side effects. SPARK 2014 is a verifiable subset of Ada 2012 [33]. The GNATprove formal verification tool of AdaCore allows to ensure reliability and correctness by performing static analysis and deductive verification. In more detail, it statically verifies programs by performing a flow analysis, and proving contracts and absence of runtime errors like division by zero, incorrect array indexing, and overflow errors. Additionally, we use annotations to support GNATprove with proof structures by inserting logical statements such as assertions and loop invariants. SPARK allows for writing *ghost code* used for specification and verification purposes without impacting the actual program's execution. In particular, ghost code can easily be removed during compilation, and SPARK guarantees that ghost code cannot impact the behavior of non-ghost code. These parts are only meant for specification and verification, and have no effect on the functional behavior of the program.

Main Contribution. We demonstrate that core concepts of propositional logic, along with modern redundancy criteria, can be expressed in SPARK, allowing us to prove their functional correctness. This in turn enables fast, verified proof checkers for LRAT unsatisfiability proofs.

2 Propositional Logic in Ada/SPARK

We briefly explain basic notions and concepts of in propositional (Boolean) logic [34]. Subsequently, we illustrate how crucial notions can be embedded into Ada/SPARK.

2.1 Syntax

Propositional formulas talk about variables that can be true/-false (propositional). Therefore, we take a set $\mathcal{V}$ of *propositional variables*. A *literal* L is either a variable A or its negation $\neg A$, which we represent by non-zero integers. The *complement* of a literal L is $\neg A$, if L is a variable; and otherwise A, if L is its negation $\neg A$. We denote the complement by $\overline{L}$ and restrict the range of literals to prohibit overflows.

```
subtype Literal_Type is Integer with
Static_Predicate =>
  Literal_Type /= 0 and then Literal_Type > -2**31 + 1
  and then Literal_Type < 2**31 - 1;
```

A *clause* C is a disjunction of literals, represented by a set of literals, and a formula F is a conjunction of clauses. Note that an empty clause can never be satisfied. We represent clauses as arrays of literals, and formulas as functional vectors:

```
type Clause_Type is array (Natural range 0 .. <>)
  of Literal_Type with
  Dynamic_Predicate =>
   Clause_Type'First = 0 and then Clause_Type'Last
                                          < 2**16;
package Clause_Vector is new
   SPARK.Containers.Functional.Vectors
  (Index_Type => Positive, Element_Type => Clause_Type);
subtype Formula_Type is Clause_Vector.Sequence;
```

2.2 Semantics

The semantics of formulas are based on the concept of interpretations. An *interpretation* I is represented by the set of propositional variables that are implicitly mapped to $true$ under I and the remaining variables (not occurring in the interpretation) to *false*.

In Ada, we represent interpretations by an array over all variables:

```
type Interpretation_Type is array (Natural
  range 0 .. 2**31 - 1) of Boolean;
```

The *model relation* $\models$ is defined as follows (for simplicity we assume symbols as used in the paragraph above): Let I be an interpretation. (i) $I \models A$ if variable $A \in I$ and (ii) $I \models \neg A$ if variable $A \notin I$. For a disjunction of literals (clause), we take $I \models (L_1 \vee \ldots \vee L_n)$ if $I \models L_i$ for some $i \in \{1, \ldots, n\}$. For a conjunction of clauses (formula), we take $I \models (C_1 \wedge \ldots \wedge C_n)$ if $I \models C_i$ for all $i \in \{1, \ldots, n\}$. The interpretation I is a *model* of the formula F if $I \models F$.

```
function Satisfies
(I : Interpretation_Type; L : Literal_Type)
return Boolean
  is (if L > 0 then I (L) else not I (-L));

function Satisfies
(I : Interpretation_Type; C : Clause_Type)
return Boolean
  is (for some L of C => Satisfies (I, L));

function Satisfies
(I : Interpretation_Type; F : Formula_Type)
return Boolean
  is (for all C of F => Satisfies (I, C));
```

If formula F has a model, F is *satisfiable*, otherwise the formula F is *unsatisfiable*. SPARK does not allow quantification over arrays, making it difficult to express the concept of satisfiability. We can address this by creating a new container type that holds all interpretations, enabling quantifiers.

```
type Interpretation_Cursor_Type  is
  record
    Overflow : Boolean;
    I : Interpretation_Type;
  end record;

type All_Interpretations is private with
  Iterable =>
    (First       => First_Interpretation,
     Next        => Next_Interpretation,
     Has_Element => Interpretation_Has_Element,
     Element     => Interpretation_Element,
     Last        => Last_Interpretation);

function Interpretations return All_Interpretations;

function Satisfiable (F : Formula_Type) return Boolean
is (for some J in Interpretations =>
      Satisfies
       (Interpretation_Element
       (Interpretations, J), F))
with Post => Satisfiable'Result
  = (for some J of Interpretations => Satisfies (J, F))
```

In particular, we can prove completeness:

Table 1: Selected Unsatisfiability Proofs for SAT Solvers

Name	Year of Publication
TraceCheck [31] A step consists of an unordered sequence of clauses such that a linear resolution derivation exists.	2014
Generalized ResolutIon Trace (GRIT) [24] A refinement of the TraceCheck format such that clauses are ordered to become unit and no search is required.	2016
Reverse Unit Propagation (RUP) [19, 20] Each step is a clause that is RUP w.r.t. the input formula and proceeding clauses.	2003
Deletion Reverse Unit Propagation (DRUP) [21] Augments RUP proofs with deletion information	2013
Deletion Resolution Asymmetric Tautology (DRAT) [15, 22] Replaces RUP by RAT in DRUP and handles formula simplification methods.	2014
LRAT [25] Augments RAT proofs with relevant clauses	2017
FRAT [26] Optional annotation of antecedents for RUP check, additional clause identifiers	2021
PR [32] Based on the notion of *witnessing* assignments that is equivalent to general redundancy criteria	2020

```
procedure Completeness
(I : Interpretation_Type)
with
  Post => (for some J of Interpretations =>
   (for all L in Interpretation_Type'Range =>
    I (L) = J (L))),
  Annotate => (GNATprove, Automatic_Instantiation),
  Ghost;
```

We relate formulas *consequence*, *equivalence*, and *equisatisfiability* relation. A formula G is a *consequence* a formula F if every model of F is a model of G. Two formulas F and G are *semantically equivalent*, in symbols $F \equiv G$, if F is a consequence of G and G is a consequence of F. Two formulas F and G are *equisatisfiable*, in symbols $F \equiv_{\mathsf{sat}} G$, if either both formulas are satisfiable or both formulas are unsatisfiable.

```
function Consequence (F, G : Formula_Type) return Boolean
is
(for all I in Interpretations =>
 (if
  Satisfies (Interpretation_Element (Interpretations, I),
                                                      F)
 then
  Satisfies (Interpretation_Element(Interpretations, I),
                                                      G)))
with Ghost;

function Equivalence (F, G : Formula_Type) return Boolean
is (Consequence (F, G) and then Consequence (G, F))
with Ghost;

function Equisatisfiable (F, G : Formula_Type)
return Boolean
is ((if Satisfiable (F) then Satisfiable (G)) and then
      (if Satisfiable (G) then Satisfiable (F)))
    with Ghost;
```

2.3 (Weak) Resolution

Let C and D be clauses such that $L \in C$ and $\overline{L} \in D$. Then, the *resolvent of C and D upon L* is the clause $(C \setminus \{L\}) \cup (D \setminus \{\overline{L}\})$. The *weak resolvent of C and D upon L* is the clause $C \cup (D \setminus \{\overline{L}\})$.

As the LRAT proof system relies on weak resolution, we restrict the presentation of this concept in SPARK:

```
-- package SAT.Weak_Resolution
function Specification
   (C : Clause_Type;
    D : Clause_Type;
    L : Literal_Type;
    R : Clause_Type)
return Boolean
is
  (for all Lit of C => Appears (R, Lit)) and then
  (for all Lit of D =>
    (if Lit /= -L then Appears (R, Lit))) and then
  (for all Lit of R =>
   (if Lit = -L then Appears (C, Lit) else
    (Appears (C, Lit) or else Appears (Dit)))) and then
  Has_No_Duplicates (R);
```

This enables us to prove that weak resolvents are consequences:

```
-- package SAT.Weak_Resolution
procedure Lemma
 (F : in Formula_Type;
  C : in Clause_Type;
  D : in Clause_Type;
  L : in Literal_Type;
  R : in Clause_Type)
ith
 Pre =>
   Clause_Vector.Last (F) < Positive'Last and then
     Contains (F, C) and then
     Contains (F, D) and then
     Appears (C, L) and then
     Appears (D, -L) and then
     Specification (C, D, L, R),
 Post => Consequence (F, Clause_Vector.Add (F, R)),
 Ghost;
```

2.4 Unit Propagation

A *unit clause* is a clause that contains exactly one element, i.e., $\{L\} \in F$ for formula F. In that case, we can safely delete all element occurrences of $\overline{L}$ in the formula and all clauses containing L (*unit rule*). Formally, the resulting formula is $G = \{C \setminus \{\overline{L}\} \mid C \in F \text{ and } L \notin C\}$. We write $F \vdash_{\mathsf{unit}} G$ to denote that G is a result of applying the unit rule in F, and $F \vdash^{*}_{\mathsf{unit}} G$ the result of repeatedly applying the *unit rule* as long as possible, i.e., the reflexive and transitive closure. This is also known as *unit propagation* [35] and crucial in modern SAT solvers.

Instead of directly modifying formulas and clauses in SPARK, we represent the unit steps as *assignments*. Therefore, we take

functional vectors over literals that contain exactly those literals that are assigned to *true* during unit propagation. Based on this idea, we leave the formula unchanged and instead add literals to an assignment.

```
-- package SAT.Unit_Reduct
function Spec
 (C : Clause_Type;
  A : Assignment_Type;
  L : Literal_Type) return Boolean
is
( -- the clause is not satisfied under A
 (for all Lp of C => not Is_True (A, Lp)) and then
  -- all literals except L are false under A
  (for all Lp of C => (if Lp /= L then Is_False (A, Lp)))
  -- L appears in C
   and then Appears (C, L)
);
```

Then, we can easily define a consequence relation based on assignments:

```
-- package SAT.Unit_Reduct
function Consequence
  (F1 : Formula_Type;
   A1 : Assignment_Type;
   F2 : Formula_Type;
   A2 : Assignment_Type)
   return Boolean
is (for all I of Interpretations =>
      (if Satisfies (I, F1) and Satisfies (I, A1)
       then Satisfies (I, F2) and Satisfies (I, A2)))
  with Ghost;
```

Finally, we can describe the relationship between two unit rule steps:

```
-- package SAT.Unit_Reduct
function Propagate
  (A  : Assignment_Type;
   Ap : Assignment_Type;
   F  : Formula_Type) return Boolean
is (for some C of F =>
      (for some L of C =>
         (Spec (C, A, L) and then
          Ap = Assignment_Vector.Add (A, L))))
with
 Pre => Assignment_Vector.Last (A) < Positive'Last,
 Ghost;
```

We can then show that a propagation step preserves equivalence with respect to the assignments:

```
procedure Propagate_Lemma
  (A  : in Assignment_Type;
   Ap : in Assignment_Type;
   F  : in Formula_Type)
with
Pre  => Assignment_Vector.Last (A) < Positive'Last
        and then Propagate (A, Ap, F),
Post => Equivalent (F, A, F, Ap),
Ghost;
```

2.5 Redundancy

We consider the two redundancy criteria *Reverse Unit Propagation (RUP)* and *Resolution Asymmetric Tautologies (RAT)*. Both are central to the modern proof systems DRAT, LRAT, and FRAT:

A clause $\{L_1, \ldots, L_n\}$ has RUP with respect to formula F if $F \cup \{\{\overline{L_i}\} \mid 1 \leq i \leq n\} \vdash^*_{\mathsf{unit}} G$ and G contains the empty clause. In this case, we know that $F \equiv F \cup \{C\}$, i.e. the addition of a RUP clause preserve equivalence.

We represent the formula $\{\{\overline{L_i}\} \mid 1 \leq i \leq n\}$ as an assignment:

```
function Assignment_Clause
  (A : Assignment_Type;
  C : Clause_Type) return Boolean
is ((for all L of C => Appears (A, -L)) and
    (for all L of A => Appears (C, -L)))
with Ghost;
```

After repeated application of the unit rule until the empty clause is detected, we know that C is RUP and is a consequence:

```
procedure Propagate_RUP_Lemma
  (A  : in Assignment_Type;
   Ap : in Assignment_Type;
   C  : in Clause_Type;
   F  : in Formula_Type)
with
 Pre =>
  Equivalent (F, A, F, Ap) and then
  (for some C of F =>
    (for all L of C => Is_False (Ap, L))) and then
  Assignment_Clause (A, C) and then
  Clause_Vector.Last (F) < Positive'Last,
 Post => Consequence (F, F.Add (C)),
 Ghost;
```

A clause C has the RAT with respect to the formula F upon the literal L , if the resolvent of C and D upon L is RUP with respect to F for all clauses $D \in F$ with $\overline{L} \in D$. In this case, it holds that $F \equiv_{\mathsf{sat}} F \cup \{C\}$. To prove the statement, we use a more general form and rely that the resolvents are consequences of F. Note that resolvents are consequences. Moreover, we restrict our attention to clauses with a certain maximum bound to simplify the resolution process.

```
-- package SAT.Strong_RAT
function Spec (F : Formula_Type;
               C : Clause_Type;
               L : Literal_Type)
               return Boolean
is
  (Appears (C, L) and then
  (for all D of F =>
     C'Length + D'Length < 2 ** 16) and then
  (for all D of F =>
    (if Appears (D, -L)
     then Consequence (F, Weak_Resolution.Resolve
                                          (C, D, L)))))
  with Ghost;
```

We can then prove two lemmas. It turns out to be beneficial to split the correctness proof in two distinct procedures:

```
procedure Lemma_Interpretation
  (F : Formula_Type;
   C : Clause_Type;
   L : Literal_Type;
   I : Interpretation_Type)
with
Pre => Spec (F, C, L) and Satisfies (I, F),
Post =>
  (Satisfies (Guard (I, L), F) and then
    Satisfies (Guard (I, L), C)) or else
  (Satisfies (Guard (I, -L), F) and then
    Satisfies (Guard (I, -L), C)),
  Annotate => (GNATprove, Automatic_Instantiation),
  Ghost;

procedure Lemma
  (F : Formula_Type;
   C : Clause_Type;
   L : Literal_Type)
with
Pre => Spec (F, C, L) and then
       Clause_Vector.Last (F) < Positive'Last,
Post => Equisatisfiable (F, F.Add (C)),
Ghost;
```

The implementation of these two lemma procedures (see Figure 1 in the Appendix) is similar to proofs found in the literature.

2.6 Proofs

A proof in our context is a sequence of clauses that has the RUP or RAT property with respect to the input formula and preceding clauses. These can be augmented with “deletion information”. Intuitively speaking, a deletion information forbids the use of previous clauses and allows the checker

to free its memory. Instead of representing these proofs as data structures in SPARK, we suggest to use a more direct approach: A proof checker usually takes a proof from a file. Thus, it is not necessary to keep the whole proof as a data structure, but can be read step-by-step.

3 Conclusion

SAT redundancy criteria serve as essential components for proof checkers, with RUP and RAT playing pivotal roles in the LRAT proof format. Our work presents a SPARK library for propositional logic, including resolution, the unit rule, RUP, and RAT. We provide several lemma procedures that incorporate these clauses maintaining equisatisfiability. Their verification works directly. Notably, some procedures, such as the resolution lemma, require no explicit proof structure at all in SPARK, allowing SMT solvers to verify them automatically. Conversely, the proof of the RAT lemma, although more challenging, successfully followed traditional proofs with the aid of SPARK's lemma library's cut-operations.

We believe that our library lays the groundwork for an efficient and verified LRAT proof checker in SPARK, with expected performance close to that of unverified C-based checkers. Although a fully verified LRAT proof checker is still under development, our initial experiments on selected instances from the SAT competition indicate comparable performance to unverified C implementations. Further optimization and comprehensive experimental evaluation remain as future work.

Acknowledgments

The work has received funding from ELLIIT funded by the Swedish government.

References

[1] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC)* (M. A. Harrison, R. B. Banerji, and J. D. Ullman, eds.), (Shaker Heights, Ohio), pp. 151–158, Assoc. Comput. Mach., New York, 1971.

[2] L. Levin, "Universal sequential search problems," *Problems of Information Transmission*, vol. 9, no. 3, pp. 265—266, 1973.

[3] A. Biere, M. Heule, H. van Maaren, and T. Walsh, eds., *Handbook of Satisfiability (2nd Edition)*. Frontiers in Artificial Intelligence and Applications, Amsterdam, Netherlands: IOS Press, Feb. 2021.

[4] J. K. Fichte, D. L. Berre, M. Hecher, and S. Szeider, "The silent (r)evolution of SAT," *Commun. ACM*, vol. 66, pp. 64–72, May 2023.

[5] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, and F. Pollitt, "CaDiCaL, Gimsatul, IsaSAT and Kissat entering the SAT Competition 2024," in *Proc. of SAT Competition 2024 – Solver, Benchmark and Proof Checker Descriptions* (M. Heule, M. Iser, M. Järvisalo, and M. Suda, eds.), vol. B-2024-1 of *Department of Computer Science Report Series B*, pp. 8–10, University of Helsinki, 2024.

[6] A. Biere, "CaDiCaL simplified satisfiability solver." `http://fmv.jku.at/cadical/`, 2019.

[7] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu, "Symbolic model checking using SAT procedures instead of BDDs," in *Proceedings of the 36th ACM/IEEE-CAS/EDAC Design Automation Conference (DAC)*, pp. 317–320, 1999.

[8] J. Rintanen, "Engineering efficient planners with SAT," in *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)* (L. D. Raedt, C. Bessière, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. J. F. Lucas, eds.), Frontiers in Artificial Intelligence and Applications, pp. 684–689, IOS Press, 2012.

[9] I. Lynce and J. Marques-Silva, "Efficient haplotype inference with Boolean satisfiability," in *Proceedings of the 21st Conference on Artificial Intelligence (AAAI)* (A. Cohn, ed.), (Menlo Park, California), pp. 104–109, The AAAI Press, 2006.

[10] N. Courtois and G. V. Bard, "Algebraic cryptanalysis of the data encryption standard," in *IMA 2007* (S. D. Galbraith, ed.), vol. 4887 of *LNCS*, pp. 152–169, Springer Verlag, 2007.

[11] J. P. Marques Silva and K. A. Sakallah, "GRASP: A search algorithm for propositional satisfiability," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 506–521, 1999.

[12] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT)* (E. Giunchiglia and A. Tacchella, eds.), vol. 2919 of *LNCS*, (Heidelberg), pp. 502–518, Springer Verlag, 2003.

[13] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *Proceedings of the 21st International Joint Conference on Artifical Intelligence (IJCAI)* (C. Boutilier, ed.), (Pasadena), pp. 399–404, Morgan Kaufmann Publishers Inc., 2009.

[14] G. Audemard, J.-M. Lagniez, B. Mazure, and L. Sais, "On freezing and reactivating learnt clauses," in *Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing (SAT)* (K. A. Sakallah and L. Simon, eds.), vol. 6695 of *LNCS*, (Heidelberg), pp. 188–200, Springer Verlag, 2011.

[15] M. Järvisalo, M. J. H. Heule, and A. Biere, "Inprocessing rules," in *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR)* (B. Gramlich, D. Miller, and U. Sattler, eds.), vol. 7364 of *LNCS*, (Heidelberg), pp. 355–370, Springer Verlag, 2012.

[16] A. Stump and D. L. Dill, "Faster proof checking in the edinburgh logical framework," in *Proceedings of the 18th International Conference on Automated Deduction (CADE-18)*, (London, UK, UK), pp. 392–407, Springer-Verlag, 2002.

[17] L. Zhang and S. Malik, "Validating sat solvers using an independent resolution-based checker: Practical implementations and other applications.," in *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pp. 10880–10885, IEEE Computer Soc., 2003.

[18] T. Jussila, C. Sinz, and A. Biere, "Extended resolution proofs for symbolic SAT solving with quantification," in *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT)* (A. Biere and C. P. Gomes, eds.), vol. 4121 of *LNCS*, pp. 54–60, Springer Verlag, 2006.

[19] E. Goldberg and Y. Novikov, "Verification of proofs of unsatisfiability for CNF formulas," in *Proceedings of the Conference on Design, Automation and Test in Europe 2003 (DATE)*, (Washington, DC, USA), pp. 10886–10891, IEEE Computer Soc., 2003.

[20] A. V. Gelder, "Verifying RUP proofs of propositional unsatisfiability," in *International Symposium on Artificial Intelligence and Mathematics (ISAIM 2008), Fort Lauderdale, Florida, USA, January 2-4, 2008*, 2008.

[21] M. J. H. Heule, W. A. Hunt Jr., and N. Wetzler, "Trimming while checking clausal proofs," in *FMCAD 2013*, pp. 181–188, IEEE, 2013.

[22] N. Wetzler, M. J. Heule, and W. A. Hunt, "DRAT-trim: Efficient checking and trimming using expressive clausal proofs," in *SAT 2014* (C. Sinz and U. Egly, eds.), vol. 8561 of *LNCS*, pp. 422–429, Springer, 2014.

[23] N. Wetzler, M. Heule, and W. A. Hunt, Jr, "Drat-trim: Efficient checking and trimming using expressive clausal proofs," in *SAT 2014* (C. Sinz and U. Egly, eds.), vol. 8561 of *LNCS*, pp. 422–429, Springer Verlag, 2014.

[24] L. Cruz-Filipe, J. Marques-Silva, and P. Schneider-Kamp, "Efficient certified resolution proof checking," *CoRR*, vol. abs/1610.06984, 2016.

[25] L. Cruz-Filipe, M. J. H. Heule, W. A. H. Jr., M. Kaufmann, and P. Schneider-Kamp, "Efficient certified RAT verification," in *CADE 2017*, vol. 10395 of *LNCS*, pp. 220–236, Springer, 2017.

[26] S. Baek, M. Carneiro, and M. J. H. Heule, "A flexible proof format for SAT solver-elaborator communication," in *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part I* (J. F. Groote and K. G. Larsen, eds.), vol. 12651 of *Lecture Notes in Computer Science*, pp. 59–75, Springer, 2021.

[27] F. Pollitt, M. Fleury, and A. Biere, "Faster LRAT checking than solving with cadical," in *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy* (M. Mahajan and F. Slivovsky, eds.), vol. 271 of *LIPIcs*, pp. 21:1–21:12, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[28] L. Cruz-Filipe, M. J. H. Heule, W. A. H. Jr., M. Kaufmann, and P. Schneider-Kamp, "Efficient certified RAT verification," in *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings* (L. de Moura, ed.), vol. 10395 of *Lecture Notes in Computer Science*, pp. 220–236, Springer, 2017.

[29] Y. K. Tan, M. J. H. Heule, and M. O. Myreen, "Verified propagation redundancy and compositional UNSAT checking in cakeml," *Int. J. Softw. Tools Technol. Transf.*, vol. 25, no. 2, pp. 167–184, 2023.

[30] Y. K. Tan, M. J. H. Heule, and M. O. Myreen, "cake_lpr: Verified propagation redundancy checking in cakeml," in *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part II* (J. F. Groote and K. G. Larsen, eds.), vol. 12652 of *Lecture Notes in Computer Science*, pp. 223–241, Springer, 2021.

[31] A. Biere, "BooleForce and TraceCheck," 2014.

[32] M. J. H. Heule, B. Kiesl, and A. Biere, "Strong extension-free proof systems," *J. Autom. Reason.*, vol. 64, no. 3, pp. 533–554, 2020.

[33] AXE Consultants, "Ada reference manual 2022 edition." `http://www.ada-auth.org/standards/22rm/RM-Final.pdf`, 2023.

[34] H. Kleine Büning and T. Lettman, *Propositional logic: deduction and algorithms*. New York, NY, USA: Cambridge University Press, Cambridge, 1999.

[35] M. Davis and H. Putnam, "A computing procedure for quantification theory," *J. of the ACM*, vol. 7, no. 3, pp. 201–215, 1960.

Appendix

```
procedure Lemma_Interpretation
     (F : Formula_Type;
      C : Clause_Type;
      L : Literal_Type;
      I : Interpretation_Type)
with
  Pre => Spec (F, C, L) and Satisfies (I, F), -- C is RAT w.r.t. F upon L
  Post =>
       (Satisfies (Guard (I, L), F) and then Satisfies (Guard (I, L), C)) or else
       (Satisfies (Guard (I, -L), F) and then Satisfies (Guard (I, -L), C)),
       Annotate => (GNATprove, Automatic_Instantiation),
   Ghost;

procedure Lemma_Interpretation
     (F : Formula_Type;
      C : Clause_Type;
      L : Literal_Type;
      I : Interpretation_Type)
   is
   begin
      if Satisfies (Guard (I, L), F) or Satisfies (Guard (I, -L), C) then
         return;
      end if;

      pragma Assert (for all D of F =>
                        (if Appears (D, -L) then
                              Consequence (F, Weak_Resolution.Resolve (C, D, L))));
      pragma Assert
        (Satisfies (Guard (I, -L), F) and then
           (for all D of F =>
                (if Appears (D, -L) then
                        Consequence (F, Weak_Resolution.Resolve (C, D, L)) and then
                        Satisfies (Guard (I, -L), Weak_Resolution.Resolve (C, D, L))
                )));

      pragma Assert (Satisfies (Guard (I, L), C));
      if Satisfies (Guard (I, L), F) then
         return;
      end if;

      pragma Assert (for some D of F => (not Satisfies (Guard (I, L), D)));
      pragma Assert (not Appears (C, -L));
      pragma Assert (for all Lp of C => not Appears (C, -Lp));

      pragma Assert (for some D of F => not Satisfies (Guard (I, L), D));

      pragma Assert
        (for all D of F =>
           (if not Satisfies (Guard (I, L), D) then
                (
                 (for all Lp of D => not Satisfies (Guard (I, L), Lp)) and then
                   (for all L of D => not Appears (D, -L)) and then
                 Appears (D, -L) and then
                       not Appears (D, L) and then
                       not Appears (C, -L) and then
                     -- I [-L] |= F, I[-L] not |= C
                 Satisfies (Guard (I, -L), Weak_Resolution.Resolve (C, D, L)) and then
                   (for some Lp of Weak_Resolution.Resolve (C, D, L) =>
                        Satisfies (I, Lp)) and then
                     (for all Lp of Weak_Resolution.Resolve (C, D, L) => Lp /= -L) and then
                   (for some Lp of Weak_Resolution.Resolve (C, D, L) =>
                        Satisfies (I, Lp) and then (Appears (C, Lp) or else Appears (D, Lp))) and then
                     (for some Lp of Weak_Resolution.Resolve (C, D, L) =>
                            Satisfies (I, Lp) and then (Appears (C, Lp) or else Appears (D, Lp))
                                                                and then Lp /= L and then Lp /= -L)
                )
           ));
      pragma Assert (False);
      end Lemma_Interpretation;
```

Figure 1: RAT Lemma Procedure showing correctness of adding RAT clauses to a formula.

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: vorstand@ada-deutschland.de
URL: ada-deutschland.de

Ada User Society

c/o Ahlan Marriott
Altweg 5
8450 Andelfingen
Switzerland
Phone: +41 52 3171975
e-mail: treasurer@ada-user.org
URL: www.ada-user.org

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 3171975
e-mail: president@ada-switzerland.ch
URL: www.ada-switzerland.ch

Ada-Europe Sponsors



46 Rue d'Amsterdam
F-75009 Paris
France
sales@adacore.com
www.adacore.com



506 Royal Road
La Caverne, Vacoas 73310
Republic of Mauritius
Contact: David Sauvage
david.sauvage@adalabs.com
www.adalabs.com



2 rue Mozart
92110 Clichy
France
Contact: Jean-Pierre Rosen
rosen@adalog.fr
www.adalog.fr/en/



Jacob Bontiusplaats 9
1018 LL Amsterdam
The Netherlands
Contact: Wido te Brake
wido.tebrake@deepbluecap.com
www.deepbluecap.com



24 Quai de la Douane
29200 Brest, Brittany
France
Contact: Pierre Dissaux
pierre.dissaux@ellidiss.com
www.ellidiss.com



3271 Valley Centre Drive,Suite 30(
San Diego, CA 92069
USA
Contact: Shawn Fanning
sfanning@ptc.com
www.ptc.com/developer-tool



Enterprise House
Baloo Avenue, Bangor
North Down BT19 7QT
Northern Ireland, UK
enquiries@sysada.co.uk
sysada.co.uk



1115 Rue René Descartes
13100 Aix en Provence
France
Contact: Patricia Langle
patricia.langle@systerel.fr
www.systerel.fr/en/



Tiirasaarentie 32
FI 00200 Helsinki
Finland
Contact: Niklas Holsti
niklas.holsti@tidorum.fi
www.tidorum.fi





Bachstrasse 51
8200 Schaffhausen
Switzerland
Contact: Ahlan Marriott
admin@white-elephant.ch
www.white-elephant.ch




Ada
europe