The journal for the international Ada community

Ada User Journal

Volume 46
Number 1
March 2025

Editorial 3
Quarterly News Digest 4
Conference Calendar 35
Forthcoming Events 44

Articles from the "3rd ADEPT Workshop: AADL by its Practitioners" of AEiC 2024

H. N. Tran et al.
ADEPT 2024 Workshop Summary 47

L. B. Becker, F. S. Gonçalves, E. F. Broering, H. A. Misson, L. Cordeiro
Safe UAV Continuous-Control Architecture Design 50

A. Calderón, I. Yarza, S. Sinisi, L. Lazzara, V. Di Valerio, G. Stazi, L. Kosmidis, M. Trompouki, A. Ulisse, A. Amonarriz, P. Onaindia
TASTE and AADL in the METASAT Model-Based Engineering Workflow 54

S. Rubini, S. Levieux, E. Cariou, F. Singhoff, H. N. Tran, G. Le Pluart
Executable AADL Models for Early System Qualification Test 58

F. Singhoff, S. Rubini, H. N. Tran, S. Levieux
AADL Modeling and Schedulability Analysis of Multiprocessor Architectures 63

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 1
March 2025

Contents

	Page
Editorial Policy for Ada User Journal	2
Editorial	3
Quarterly News Digest	4
Conference Calendar	35
Forthcoming Events	44
Articles from the "3rd ADEPT Workshop: AADL by its Practitioners" of AEiC 2024	
H. N. Tran et al. "*ADEPT 2024 Workshop Summary*"	47
L. B. Becker, F. S. Gonçalves, E. F. Broering, H. A. Misson, L. Cordeiro *"Safe UAV Continuous-Control Architecture Design"*	50
A. Calderón, I. Yarza, S. Sinisi, L. Lazzara, V. Di Valerio, G. Stazi, L. Kosmidis, M. Trompouki, A. Ulisse, A. Amonarriz, P. Onaindia *"TASTE and AADL in the METASAT Model-Based Engineering Workflow"*	54
S. Rubini, S. Levieux, E. Cariou, F. Singhoff, H. N. Tran, G. Le Pluart *"Executable AADL Models for Early System Qualification Test"*	58
F. Singhoff, S. Rubini, H. N. Tran, S. Levieux *"AADL Modeling and Schedulability Analysis of Multiprocessor Architectures*	63
Ada-Europe Associate Members (Ada Organizations)	72
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

Over the past few years, I've usually started my editorials by highlighting important events or situations of special relevance to the Ada community. This time, I would like to turn inward and take a moment to recognize the work of the Ada User Journal editorial team. I am sure most of our readers are aware of the dedication these colleagues put into creating, gathering, and editing a new issue every three months, some of them having been doing this already for many years. Still, this entirely voluntary work may sometimes go unnoticed, so I think it is worth giving it the visibility it deserves. You will find their names on the inside cover of the journal. Kudos to them!

As for the technical contents, this issue features a set of articles from the 3rd ADEPT Workshop, co-located with AEiC 2024, in Barcelona, Spain. The set of articles begins with a summary of the workshop, outlining its scope and program. Of the nine technical presentations given at the workshop, whose slides may be found on the workshop web page, four have been further consolidated into full articles that are included in this issue. A common aspect of all articles is the use of AADL (Architecture Analysis and Design Language) for modelling both hardware and software architectures. Becker et al. apply AADL to modelling the software architecture of UAVs, Calderón at al. use AADL within a model-based engineering workflow in the context of the METASAT project, Rubini et al. use AADL models as input to a SystemC simulator for analysing the temporal behaviour of the simulated system, and Singhoff et al. employ AADL to model and carry out schedulability analysis of multiprocessor systems.

The News Digest section, prepared by Alejandro R. Mosteo, is slightly longer than usual, reflecting the inclusion of news from a new community-run Discourse forum. The Conference Calendar and Forthcoming Events sections, prepared by Dirk Craeynest, list all the possibly relevant events for the Ada community. And I would like to highlight the Call for Participation in the 29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025), which will take place in Paris next June, included in pages 44-45.

Antonio Casimiro
Lisboa
March 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	4
Ada-related Events	4
Ada-related Resources	8
Ada-related Tools	13
Ada and Other Languages	22
Ada Practice	23

[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,

For a while it has been increasingly apparent that traffic has been migrating from the venerable Usenet group comp.lang.ada [1] to a new community-run Discourse forum [2] (and other locations you can find in our Social Media tracker [3]). To maintain the relevance of the News Digest, I am happy to announce that, starting with this issue, this new forum is also captured in the Digest, as the most similar medium to the traditional newsgroup.

While certainly there is some overlap in the concurrence of both venues, they also have their own distinct character (and many more newbies at the new forum) that I hope will enrich the News Digest going forward.

Sincerely,

Alejandro R. Mosteo.

[1] https://usenet.ada-lang.io/comp.lang.ada/

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

[3] "Ada on Social Media", in Ada-related Resources.

Ada-related Events

Ada-Europe Conference - Updated Call for Contributions

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe conference - 7 Feb Journal Track Extended Deadline
Date: Sun, 12 Jan 2025 16:50:30 +0000
Newsgroups: comp.lang.ada

UPDATED Call for Contributions

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

10-13 June 2025, Paris, France
www.ada-europe.org/conference2025

*** Journal-track deadline EXTENDED to 7 February 2025 ***

*** Other submissions by 24 February 2025 ***

Organized by Ada-Europe and Ada-France

#AEiC2025 #AdaEurope #AdaProgramming

General Information

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

- Journal track papers present research advances supported by solid theoretical foundation and thorough evaluation.
- Industrial track contributions highlight the practitioners' side of a challenging case study or industrial project.
- Work-in-progress track papers illustrate novel research ideas 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.
- Vendor presentations and exhibitions allow for companies to showcase their latest products and services.

Schedule

7 February 2025: EXTENDED submission deadline for journal track papers

24 February 2025: Deadline for submission of industrial track papers, work-in-progress papers, and tutorial and workshop proposals

28 March 2025: First round notification for journal track papers, and notification of acceptance for all other types of submissions

10-13 June 2025: Conference

Scope and Topics

The conference is a leading international forum for providers, practitioners, and researchers in reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the design, development, and maintenance of long-lived, high-quality software systems for a challenging variety of application domains. The program will allow ample time for keynotes, Q&A sessions, discussions, and social events. Participants include practitioners and researchers from industry, academia, and government organizations active in the promotion and development of reliable software technologies.

The topics of interest for the conference include but are not limited to (more specific topics are described on the conference web page):

- Formal and Model-Based Engineering of Critical Systems;
- High-Integrity Systems and Reliability;
- AI for High-Integrity Systems Engineering;
- Real-Time Systems;
- Ada Language;
- Applications in relevant domains.

More specific topics are described on the conference web page, at www.ada-europe.org/conference2025.

Call for Journal Track Submissions

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

JSA has adopted the Virtual Special Issue model to speed up the publication process, where Special Issue (SI) papers are published in regular issues, but marked as SI papers. Acceptance decisions are made on a rolling basis. Therefore, authors are encouraged to submit papers early, and need not wait until the submission deadline. Authors who have successfully passed the first round of review will be invited to present their work at the conference. The abstract of the accepted contributions will be included in the conference booklet.

Please note that the Ada-Europe organization will waive the Open Access fees for the first four accepted papers, which do not already enjoy OA from other agreements with the Publisher. Subsequent papers will follow JSA regular publishing track. Prospective authors may direct all enquiries regarding this track to the corresponding chairs, Laurent Pautet and Sara Royuela.

Call for Industrial Track Submissions

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

The abstracts of the accepted contributions will be included in the conference booklet. The corresponding authors will get a presentation slot in the prime-time technical program of the conference and will also be invited to expand their contributions into full-fledged articles for publication in the Ada User Journal, which will form the proceedings of the industrial track of the Conference.

Prospective authors may direct all enquiries regarding this track to its chairs, Daniela Cancila and Laurent Gouzenes.

Call for Work-in-progress Track Submissions

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

The 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 offered the opportunity to expand their contributions into 4-page articles for publication in the Ada User Journal, which will form the proceedings of the WiP track of the conference.

Prospective authors may direct all enquiries regarding this track to the corresponding chairs, Hai Nam Tran and Anish Bhobe.

Award

The Ada-Europe organization will offer an honorary award for the best technical presentation, to be announced in the closing session 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 24 February 2025 to the respective chair, Robert Cholay, by email, with subject line: "[AEiC 2025: tutorial proposal]".

The authors of accepted full-day tutorials will receive a complimentary conference registration, halved for half-day tutorials. The Ada User Journal will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

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

Academic Listing

The Journal of Systems Architecture, publication venue of the journal-track proceedings of the conference, is Q1 ranked, with CiteScore 8.5 and Impact Factor 3.7. The Ada User Journal, venue of all other technical proceedings of the conference, is indexed by Scopus and by EBSCOhost in the Academic Search Ultimate database.

Call for Exhibitors and Sponsors

The conference will include a vendor and technology exhibition, with the option of a 20 minutes presentation as part of the conference program. Interested providers should direct inquiries to the Exhibition & Sponsorship Chair, Ahlan Marriott.

Venue

The conference will take place at Mines Paris. Mines Paris - PSL, a founding member of Université PSL, is a leading French engineering school and the French leader institution in research partnerships. Founded 240 years ago to help spur the energy efforts called by the Industrial Revolution, it has been since training engineers in a wide spectrum of scientific disciplines. With about 1,500 students, including 100 PhD graduates per year, Mines Paris - PSL hosts 18 research

centers and 5 academic departments. It is located along the Luxembourg gardens, next to the Quartier Latin, and is close to public transportation, including line B of the RER to Charles De Gaulle airport (CDG).

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

Organizing Committee

- Conference Chair

Jean-Pierre Rosen, Adalog & Ada-France
rosen at adalog.fr

- Journal track Co-chairs

Laurent Pautet, Telecom Paris
laurent.pautet at telecom-paris.fr

Sara Royuela, Barcelona Supercomputing Center
sara.royuela at bsc.es

- Industrial track Co-chairs

Daniela Cancila, CEA LIST
daniela.cancila at cea.fr

Laurent Gouzenes, Pacte-Novation
lgouzenes at pactenovation.fr

- Work-in-progress track Co-chairs

Hai Nam Tran, University of Brest
hai-nam.tran at univ-brest.fr

Anish Bhobe, Telecom Paris
anish.bhobe at telecom-paris.fr

- Tutorial Chair

Robert Cholay, Systerel
robert.cholay at systerel.fr

- Workshop Chair

Anish Bhobe, Telecom Paris
anish.bhobe at telecom-paris.fr

- Exhibition & Sponsorship Chair

Ahlan Marriott, White Elephant GmbH
ahlan at ada-switzerland.ch

- Finance Chair

Paul Duquennoy
paul.duquennoy at free.fr

- Publicity Chair

Dirk Craeynest, Ada-Belgium & KU Leuven
dirk.craeynest at cs.kuleuven.be

- Local Chair

Pierre Jouvelot, Mines Paris, PSL University
pierre.jouvelot at minesparis.psl.eu

- Webmaster

Hai Nam Tran, University of Brest
hai-nam.tran at univ-brest.fr

Journal Track Committee

Alejandro Mosteo, CUD Zaragoza, Spain
Andrea Marongiu, University of Modena and Reggio Emilia, Italy
Ann Laguna, De La Salle University, USA
Angeliki Kritikakou, University of Rennes, France
António Casimiro, University of Lisbon, Portugal
Bjorn Andersson, SEI, USA
C. Michael Holloway, NASA, USA
Cristina Seceleanu, Mälardalen University, Sweden
Doug Schmidt, Vanderbilt University, USA
Frank Singhoff, University of Brest, France
George Lima, Universidade Federal da Bahia, Brazil
Isaac Amundson, Rockwell Collins, USA
Jérôme Hugues, CMU/SEI, USA
John B Goodenough, CMU, USA
Kerstin Bach, Norwegian University of Science and Technology, Norway
Kristoffer Nyborg Gregertsen, SINTEF Digital, Norway
Laura Carnevali, University of Florence, Italy
Laurent Pautet, Telecom ParisTech, France
Leonidas Kosmidis, Barcelona Supercomputing Center, Spain
Liliana Cucu-Grosjean, Inria, France
Luis Miguel Pinho, ISEP & INESC TEC, Portugal
Mario Aldea Rivas, University of Cantabria, Spain
Martina Maggio, Saarland University, Germany
Matthias Becker, KTH - Royal Institute of Technology, Sweden
Patricia López Martínez, University of Cantabria, Spain
Risat Pathan, Chalmers University, Sweden
Sara Royuela, Barcelona Supercomputing Center, Spain
Sergio Sáez, Universitat Politècnica de València, Spain
Susanne Graf, University Grenoble Alpes, France
Steven Xiaotian Dai, University of York, UK
Tucker Taft, AdaCore, USA
Tullio Vardanega, University of Padua, Italy
Xhevahire Ternava, Telecom Paris, France

Industrial Track Committee

Alexander Viehl, FZI Research Center for Information Technology, Germany
Ana Rodríguez, Silver Atena, Spain
Claire Dross, AdaCore, France
Daniela Cancila, CEA, France
Elena Lisova, Volvo CE, Sweden
Enricco Mezzetti, Barcelona Supercomputing Center, Spain
Federico Aromolo, Scuola Superiore Sant'Anna, Italy
Helder Silva, Edisoft, Portugal
Hugo Torres Vieira, Evidence Srl, Italy
Irune Agirre, Ikerlan, Spain
José Ruiz, AdaCore, France
Laurent Gouzènes, Pacte Novation, France
Marco Panunzio, Thales Alenia Space, France
Michael Pressler, Robert Bosch GmbH, Germany
Raúl de la Cruz, Collins Aerospace, Ireland
Santiago Urueña, GMV, Spain
Saqib Hasan, Collins, USA, USA

Work-in-Progress Track Committee

Abderaouf Nassim Amalou, University of Nantes, LS2N, France
Alan Oliveira, University of Lisbon, Portugal
Alexandre Honorat, INRIA Grenoble, France
Anish Bhobe, Telecom Paris, France
Audrey Quedet, University of Nantes, LS2N, France
Hai Nam Tran, University of Brest, Lab-STICC, France
J. Javier Gutiérrez, University of Cantabria, Spain
Jérémie Guiochet, LAAS-CNRS, France
José Cecílio, University of Lisbon, Portugal
Kalinka Branco, University of São Paulo, Brazil
Leandro Buss Becker, University of Manchester, UK
Samuel Tardieu, Telecom Paris, France
Sara Abbaspour, Mälardalen University, Sweden
Shuo-Han Chen, NYCU, Taiwan
Stéphane Rubini, University of Brest, Lab-STICC, France
Tiago Carvalho, ISEP, Portugal

Previous Editions

Ada-Europe organizes annual international conferences since the early 80's. This is the 29th event in the Reliable Software Technologies series, previous ones being held at Montreux, Switzerland ('96), London, UK ('97), Uppsala, Sweden ('98), Santander, Spain ('99), Potsdam, Germany ('00), Leuven, Belgium ('01), Vienna, Austria ('02),

Toulouse, France ('03), Palma de Mallorca, Spain ('04), York, UK ('05), Porto, Portugal ('06), Geneva, Switzerland ('07), Venice, Italy ('08), Brest, France ('09), Valencia, Spain ('10), Edinburgh, UK ('11), Stockholm, Sweden ('12), Berlin, Germany ('13), Paris, France ('14), Madrid, Spain ('15), Pisa, Italy ('16), Vienna, Austria ('17), Lisbon, Portugal ('18), Warsaw, Poland ('19), online from Santander, Spain ('21), Ghent, Belgium ('22), Lisbon, Portugal ('23), and Barcelona, Spain ('24).

Information on previous editions of the conference can be found at www.ada-europe.org/confs/ae.

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)

* June 10-13, 2025, Paris * France, www.ada-europe.org/conference2025

(V3.1)

12th Ada DevRoom, FOSDEM 2025

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: 12th Ada DevRoom @ FOSDEM 2025, Sun 2 Feb, Brussels & online
Date: Wed, 29 Jan 2025 11:05:41 +0000
Newsgroups: comp.lang.ada

Reminder: 12th Ada Developer Room on Sunday 2 February 2025 at FOSDEM 2025 in Brussels and online! (09:00-12:50 CET)

Call for Participation

12th Ada Developer Room at FOSDEM 2025

Sunday 2 February 2025, Brussels, Belgium

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

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

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

#AdaFOSDEM #AdaDevRoom #AdaProgramming

#AdaBelgium #AdaEurope #FOSDEM2025

FOSDEM [3], the Free and Open source Software Developers' European Meeting, is a non-commercial two-day weekend event organized early each year in Brussels, Belgium. It is highly developer-oriented and brings together 8000+ participants from all over the world. The 2025 edition takes place on Saturday 1 and Sunday 2 February. It is free to attend and no registration is necessary.

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

Program overview:

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

The Ada at FOSDEM 2025 web-page has all details, such as the full schedule, abstracts of presentations, biographies of speakers, and pointers to more info, including live video streaming and recordings afterwards. For the latest information at any time, contact Fernando Oleo Blanco <irvise@irvise.xyz> or see: www.cs.kuleuven.be/~dirk/ada-belgium/events/25/250202-fosdem.html

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

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

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

Dirk Craeynest, FOSDEM Ada DevRoom team

Dirk.Craeynest@cs.kuleuven.be
Dirk.Craeynest@kuleuven.be

(V20250129.1)

2024 Crate of the Year Awards!

From: Irvise <Irvise@forum.ada-lang.io>
Subject: 2024 Crate of the Year Awards!
Date: Mon, 3 Feb 2025 20:51:08 +0000
Newsgroups: forum.ada-lang.io

The winners of the crate of the year have been announced! Ada/SPARK Crate Of The Year 2024 Winners Announced! [1]

I am specially interested in the embeddable LISP!

Also congratulations to @LionelDraghi and @kevlar700

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

From: LionelDraghi <LionelDraghi@forum.ada-lang.io>
Date: Mon, 3 Feb 2025 22:11:37 +0000

First, many thanks to AdaCore for the prize, and congrats to my fellow co-winners!

I'm trying to get some visibility outside of the Ada world. (bbt is not specific to Ada)

Because, first, I need more user's feedback, and second, I would be happy if an Ada tool was used outside our world.

I posted a short (but attention-grabbing!) message on DEV Community [1]. It seems to be a complete fail.

There is one post per minute, after two hours mine is already relegated to the thens page, with a glorious zero comment / zero like.

Have you any idea of places that may be interested in testing/documentation/TDD BDD where I could drop a message?

[1] https://dev.to/lioneldraghi/my-dream-way-of-testing-8m9

From: mgrojo <mgrojo@forum.ada-lang.io>
Date: Tue, 4 Feb 2025 21:57:53 +0000

Congratulations, @LionelDraghi, @kevlar700 and Brent Seidel!

> Have you any idea of places that may be interested in testing/documentation/TDD BDD where I could drop a message?

I just published your article on Hacker News [1]. It's difficult to get traction there, but there's nothing to lose in trying.

By the way, bbt has been very useful and easy to set up for testing my latest project

I've just set public the repository, and you can see here [2] how I'm using it.

[1] https://news.ycombinator.com/item?id=42939189

[2] https://github.com/mgrojo/coap_spark/blob/c326e1c188bdfb88ab5f815a10779f5745ab32a8/client/tests/coap_client_tests.md

Ada Monthly Meetup, 1st of March 2025

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Subject: Ada Monthly Meetup, 1st of March 2025
Date: Sat, 15 Feb 2025 08:17:39 +0000
Newsgroups: comp.lang.ada

I would like to announce the March (2025) Ada Monthly Meetup which will be taking place on the 1st of March at 14:00 UTC time (15:00 CET). As always, the meetup will take place over at Jitsi [1]. The Meetup will also be livestreamed/recorded to Youtube.

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

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

I will talk a bit about FOSDEM, how it went and the overall experience. I will also talk about the Ada Developers Workshop, which will take place during AEiC 2025 [2], which we still need to make public but we have been accepted officially.

P.S: you can see the December summary in Ada Monthly Meeting December 2024 [3]

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

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

[3] https://forum.ada-lang.io/t/ada-monthly-meetup-7th-december-2024/1444/12

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Date: Sat, 1 Mar 2025 23:53:44 +0000

The minutes of the meeting are available in https://forum.ada-lang.io/t/ada-monthly-meetup-1st-of-march-2025/1815/5?u=irvise

Ada Developers Workshop, AEiC 2025

From: Irvise <Irvise@forum.ada-lang.io>
Subject: Ada Developers Workshop @ AEiC 2025 - Paris, June 13th
Date: Sun, 2 Mar 2025 10:41:35 +0000
Newsgroups: forum.ada-lang.io

It pleases me to announce the second edition of the Ada Developers Workshop [1], which will be taking place within the wider Ada-Europe international Conference on Reliable Software Technologies, Paris 10-13th of June.

The Workshop will be taking place on Friday the 13th, this way we minimize the amount of free days people may need to take off from work and also have the weekend to enjoy Paris or the rest of France.

Please, read the Workshop's website to know more about the submission requirements, the deadlines and contact information. Nonetheless, we would like to schedule technical presentations, tutorials, demos, live performances, project status reports, discussions, etc, in the Ada Developers Workshop.

The cost to attend the workshop is still to be decided, but we expect it to be similar to last year, so a highly reduced cost is to be expected, that way we can maximize in-person attendance. We will also be featuring a live-stream just like last year.

Best regards,
The Ada Developers Workshop team (@DirkCraeynest, @Fabien.C and myself)

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

Ada Monthly Meetup, 5th of April 2025

From: Irvise <Irvise@forum.ada-lang.io>
Subject: Ada Monthly Meetup, 5th of April 2025
Date: Wed, 19 Mar 2025 21:49:53 +0000
Newsgroups: forum.ada-lang.io

I would like to announce the April (2025) Ada Monthly Meetup which will be taking place on the 5th of April at *13:00 UTC time (15:00 CEST)*. As always, the meetup will take place over at Jitsi. The Meetup will also be livestreamed/recorded to Youtube.

Important: due to the change of Timezones in most of Europe (from CET to CEST), the meetup will take place at the same time in Europe, but it may change in other parts of the world!

Also, I will not organise a May meetup as I will not be available. If someone wants to step up, feel free to take over!

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

Here are the connection details from previous posts:

The meetup will take place over at Jitsi, a conferencing software that runs on any modern browser. The link is Jitsi Meet [1] The room name is "AdaMonthlyMeetup" and in case it asks for a password, it will be set to "AdaRules".

I do not want to set up a password, but in case it is needed, it will be the one above without the quotes. The room name is generally not needed as the link should take you directly there, but I want to write it down just in case someone needs it.

I will talk about the Ada Developers Workshop, which will take place during AEiC 2025 [2], for which we are still accepting proposals!

Best regards and see you soon!

P.S: you can see the December summary in Ada Monthly Meeting March 2025 [3]

P.S: feel free to repost this in other forums or chats, such as Reddit! >:D

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

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

[3] https://forum.ada-lang.io/t/ada-monthly-meetup-1st-of-march-2025/1815/5

Ada-related Resources

[Delta counts are from January 29th to June 23rd. Tabulated 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: 23 Jun 2025 20:10 CET
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: 9_180 (+188) members [1]
- LinkedIn: 3_654 (+74) members [2]
- Stack Overflow: 2_443 (+8) questions [3]
- Ada-lang.io: 382 (+61) users [4]
- Gitter: 289 (+12) people [5]
- Telegram: 227 (+13) users [6]
- Discord: 191 (new) users [7]
- Libera.Chat: 75 (-3) concurrent users [8]

[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

[8] 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: 23 Jun 2025 20:10 CET
To: Ada User Journal readership

GitHub: >373_000* (new) repositories [1]
>1_000* (=) developers [1]

Alire: 1_425 (+110) releases [2]
557 (+36) crates [3]

Rosetta Code: 1_024 (+16) examples [4]
43 (+1) developers [5]

Sourceforge: 242 (=) projects [6]

Open Hub: 14 (=) projects [7]

Codelabs: 63 (+2) 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: 23 Jun 2025 20:10 CET
To: Ada User Journal readership

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

- TIOBE Index: 11 (+15)
1.70% (+1.05%) [1]

- PYPL Index: 15 (=) 1.21% (+0.08%) [2]

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

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

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

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

- Languish Trends: 153 (-12)
0.01% (=) [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/

AdaCore's Training Material

From: Irvise <Irvise@forum.ada-lang.io>
Subject: Add AdaCore's Training material to Ada-Lang
Date: Sun, 12 Jan 2025 17:28:06 +0000
Newsgroups: forum.ada-lang.io

AdaCore maintains a repo with quite a lot of Ada training material [1]. Its contents are licensed under CC BY 4.0. This repo IS DIFFERENT from that of https://learn.adacore.com/. The repo is mostly composed of tons of slides regarding the different parts of Ada in self-contained units. I see that @ohenley is one of the main contributors/maintainers. I am pinging @Fabien.C, @pyj and @JeremyGrosser too just for good measure.

I had known of its existence since… a long long time. However, I had never actually taken a good look at the contents until yesterday. And oh, boy, what a gold mine! It has some really high quality slides in a somewhat bite-sized extension. They are down to the point, readable, clean, and with straight to the point examples. They are great. And I think we ought to have them in one way or another in the Learn page of Ada-Lang.

I believe a lot of people do not know of the existence of these resources, as I never saw them mentioned anywhere! I think this is one of the missing key points of literature/learning resources of Ada. Simple, broken down examples with a short explanation text. So. What do you think? Should we maybe have a section under Learn that lists these slides? We should have a system that automatically downloads their latest versions and puts them in the server in an ordered and discoverable manner for people to easily find them.

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

From: jere <jere@forum.ada-lang.io>
Date: Sun, 12 Jan 2025 18:13:43 +0000

I think having some way to find them from ada-lang is a good idea. I do also think that however we do it, it should be easier to go directly to the slides so a new person doesn't have to sit there going through folders to find the good stuff.

From: Irvise <Irvise@forum.ada-lang.io>
Date: Sun, 12 Jan 2025 20:14:33 +0000

Yes! The contents would have to be organised in a way that people can quickly find what they want

And I was thinking just right now… I had in mind to put the PDFs in Ada-Lang and let people read those (they are processed through LaTeX and the quality shows) but… I just came to the realisation that they are written in RST format… which could be somewhat easily transformed to Markdown and then be processed by the build infra (docusaurus) from Ada-Lang… Mmmmm… Maybe we could have our own native pages for them… Mmmm…

From: leogermond <leogermond@forum.ada-lang.io>
Date: Tue, 14 Jan 2025 11:02:46 +0000

I'm a maintainer for this material. First off, thanks for the kind words, we put lots of efforts into this content, so that this shows is very cool.

The material can certainly be turned into md files using pandoc but take care of two things:

- We are using custom directives for e.g. quizzes, and as a result we have a pandoc/beamer_filter.py script that converts these into LaTeX (our own target file format).
- These are used through beamer for LaTeX, so the content might be formatted into a somewhat wonky markdown format.

You can use and take inspiration from the pandoc/pandoc_fe.py script that is currently our wrapper around pandoc for RST → tex → PDF conversion.

Let me know how that goes, PRs are definitely welcome!

From: Irvise <Irvise@forum.ada-lang.io>
Date: Tue, 14 Jan 2025 22:37:06 +0000

Thank you for the work put in the content.

Yes, I was thinking about using pandoc to run the translation from RST to MD. I am aware of pandoc's and each format's limitations. I was wondering how you actually managed to get such nicely formatted Beamer slides with just RST, but your explanation demystifies it.

I think we have two possible paths here…

- Write in JS/TS some procedures that will be triggered during the build of the website. These triggers would call pandoc on the RST files from the repo and process them so that Docusaurus can ingest them in a controlled manner.
- This would be an automatic system. With the drawback of pulling pandoc as a build dependency and requiring a bit of work from our side.
- Have your files preprocessed externally, such as a manually run set of steps or a script and pull the results into our repo.
- This would be less work, not require pandoc for Docusaurus but it would not be as automated as the other system.

I will open a PR in the Ada-Lang repo to track these ideas

Ada Developers: How Do You Find Opportunities?

From: andreamancuso <andreamancuso@forum.ada-lang.io>
Subject: Ada Developers: How Do You Find Opportunities?
Date: Fri, 7 Mar 2025 19:07:32 +0000
Newsgroups: forum.ada-lang.io

I've been diving deeper into Ada and other niche languages, and one thing I noticed is how tough it can be to find job opportunities for Ada developers.

I created *beyond-tabs.com [1]* as a *non-profit* initiative to help connect developers with job listings in less mainstream languages, including Ada. My goal is to make it easier for developers to find relevant opportunities, not to profit from it.

I wanted to ask: how do you all approach finding jobs or freelance opportunities in the Ada ecosystem? Do you feel there are enough resources, or do you think there's a gap in visibility? I'm all ears for any advice or thoughts on the matter!

[1] https://beyond-tabs.com

From: waleedmebane <waleedmebane@forum.ada-lang.io>
Date: Sat, 8 Mar 2025 00:41:33 +0000

That looks like a great site. Thanks for that resource!

I've been working as a research software engineer using mostly OCaml and C and some C# for the last 3.5 years, but the research funding is coming to an end, and I expect to be looking for a job toward the end of this year.

I am new to Ada, and I've never had a job as an Ada software engineer. I'd previously mostly been a C++ software engineer, so I'm looking forward to seeing what others will say about how they find Ada jobs.

From: waleedmebane <waleedmebane@forum.ada-lang.io>
Date: Sat, 8 Mar 2025 00:59:28 +0000

I've done some searches to try to see what the landscape is like of Ada jobs. Not too many seem to come up with the search terms "Ada" and "Spark", but I've also tried "automotive", "aviation", and "formal verification". Since I also have some background with AI, I've looked for (semi-)automated vehicle industry listings. There is an acronym that is not coming to mind right now. I saw a couple from Nvidia that are not listed at your site. I also read about Ada on discussion sites and HackerNews, and I came across mention that Toyota is switching to Ada.

When I get ready to and need to search for a job in earnest later this year, according to my expectation, I plan to look at the websites of companies listed as users of Ada on the AdaCore website.

From: andreamancuso <andreamancuso@forum.ada-lang.io>
Date: Sat, 8 Mar 2025 08:27:53 +0000

Thank you very much for the feedback and for sharing your experience. I'll dig those Nvidia roles.

Best of luck with your job search when the time comes.

From: andreamancuso <andreamancuso@forum.ada-lang.io>
Date: Sat, 8 Mar 2025 09:59:24 +0000

I could not find Ada roles at Toyota; I will keep looking.

Out of the 3 Nvidia roles mentioning Ada, only one, an internship, sounds Ada-heavy. The other two are more like C/C++ roles with some Ada. As such, I am only listing the internship for now.

From: zertovitch <zertovitch@forum.ada-lang.io>
Date: Sat, 8 Mar 2025 11:03:05 +0000

> I wanted to ask: how do you all approach finding jobs or freelance opportunities in the Ada ecosystem?

Through an Ada association (Ada Switzerland), for a job /and/ freelance opportunities.

From: DirkCraeynest <DirkCraeynest@forum.ada-lang.io>
Date: Mon, 10 Mar 2025 09:02:03 +0000

Similarly, those interested in Ada jobs in Belgium find info via the Ada-Belgium organization and via its "Ada Jobs" webpages at Ada Jobs [1]

Mind you, many Ada-related positions are often not submitted to job websites, but are filled via "word of mouth" channels…

[1] https://people.cs.kuleuven.be/~dirk.craeynest/ada-belgium/jobs/

Is alire.ada.dev Maintained?

From: pchapin <pchapin@forum.ada-lang.io>
Subject: Is alire.ada.dev maintained?
Date: Thu, 13 Mar 2025 21:46:30 +0000
Newsgroups: forum.ada-lang.io

Two days ago there was an announcement on the Reddit Ada sub that Alire 2.1.0 has been released. In fact, on ada-lang.io the download link for Alire offers version 2.1.0. However, on alire.ada.dev, the main download link still points to version 2.0.2. This leads me to wonder if the alire.ada.dev site is being actively maintained, or what the story might be. The alire.ada.dev site feels like the main site for Alire, yet it also seems out of date. Is ada-lang.io a more appropriate resource for Alire material?

From: jere <jere@forum.ada-lang.io>
Date: Thu, 13 Mar 2025 22:46:44 +0000

The github repo for the site is https://github.com/alire-project/alire.ada.dev

It's maintained, but I would bet that the maintainer is just busy (not a paid job or team associated with Alire). If you can't wait, you can try doing a pull request with the changes and that might speed it up (and I am sure the maintainer would welcome any help he can get).

From: pchapin <pchapin@forum.ada-lang.io>
Date: Fri, 14 Mar 2025 11:46:02 +0000

I can appreciate that volunteers have limited resources. It's nice that the site accepts pull requests. Still, it seems weird to me for the release to be announced in one venue (Reddit in this case) /before/ it is actually available on the official site.

Several years ago I was the maintainer of an open source project. That meant I was responsible for making releases happen. I had a checklist. After building and testing the installers, one of the steps was to make the release available on the official channels /and then/ announce it to the world. The race condition was if someone noticed the release was available before it was announced, which seems far better than if it is announced before it is available.

What happened in this case makes the community seem fractured and uncoordinated. Don't the people who create the releases talk to the people who run the web site? Why should a random community member make a pull request to update the site to reflect the release? Shouldn't the people who create the release do that? As an outsider, it looks messy.

To put it a different way: I'm working with some students this summer on an Ada project. They are new to Ada, so I'm planning to give them a tutorial over the

coming weeks. I want to start by saying, "Go to alire.ada.dev and download the latest version of Alire." However, that doesn't work! Instead I have to say, "Go to ada-lang.io and download the latest version of Alire, but use alire.ada.dev for the documentation." They are naturally going to ask, "Why is it like that?" How do I answer them?

My students are currently in a Rust class, but are interested in learning about Ada. The Rust community seems far more coordinated than the Ada community. This issue… the very first exposure the students will have to Ada… doesn't create a good impression.

From: pyj <pyj@forum.ada-lang.io>
Date: Sat, 15 Mar 2025 05:11:44 +0000

> What happened in this case makes the community seem fractured and uncoordinated.

The biggest champion of community building is @Irvise, with his vision being a multi-year effort. It seems to be working with the return of the half-day Ada Dev Room at FOSDEM and the forum here being more active.

The community and available resources are spread out among many different parties. ada-lang.io [1] is a community site, there's not a real maintainer, though @JeremyGrosser keeps it and these forums running.

Most of the open source community seems to revolve around GNAT, since that's freely available. The AdaCore folks seem to do their own thing, which makes sense since they're paid to support clients of their stack based on GNAT, but they make a lot of things available to the community and do respond to GitHub issues. They also run learn.adacore.com [2], which is a great tutorial site, and you see them here from time to time. I'm not sure what the story is with Alire.

> This issue… the very first exposure the students will have to Ada… doesn't create a good impression.

It's not great yet, but the situation has remarkably improved since I started writing Ada in mid 2021. I saw where it was and it's a night and day difference if you want to write open source Ada.

Since 2021, we've seen Alire 1.0, significantly improved VS Code support, learn.adacore.com [2] improvements, a new code formatter released (GNATformat), ada-lang.io [1], AdaCore licensing changes to make libraries more permissive [3], this forum started, a new ARG page [4], multiple "package of the year" competitions, toolchain installation with Alire and associated cleaning up of toolchain issues, gnatprove runnable with Alire, there's now a Ada community meeting most months, https://www.getada.dev/ [5], and probably more stuff that I'm forgetting about.

We still need more varied tutorials, walkthroughs and guides.

[1] http://ada-lang.io

[2] http://learn.adacore.com

[3] https://blog.adacore.com/a-new-era-for-ada-spark-open-source-community

[4] https://arg.adaic.org/ada-reference-manual

[5] https://www.getada.dev/

From: jere <jere@forum.ada-lang.io>
Date: Sat, 15 Mar 2025 16:39:05 +0000

I believe getadanow was one of a few websites that David Botton made around the time when he released the original Gnoga browser GUI library. He also made learnadanow which I don't think exists anymore, but back in the day (2014ish maybe) he tried to get a lot of the community excited about ada and tried to get a lot more resources out there (mostly just comp.lang.ada and IRC available at the time, so he was trying to improve). The humming bird mascot came out of some of that effort (along with some other community folks).

I don't know who the maintainer of getadanow is offhand, but David may have handed it to someone (Blady took over Gnoga, LearnAdaNow disappeared I think).

EDIT: Looks like Gnoga was 2014 and the websites and mascot were in 2015.

Gnoga Announce:
https://groups.google.com/g/comp.lang.ada/c/R1lmoQj6GtY

Websites:
https://groups.google.com/g/comp.lang.ada/c/Jx4nadPPxtM/m/mGO-txWVso4J

Mascot:
https://groups.google.com/g/comp.lang.ada/c/VDERgHuRkdw/m/IlphRXw9LFUJ

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

From: Irvise <Irvise@forum.ada-lang.io>
Subject: [GSoC & Ada] Working on open source Ada projects and getting paid?
Date: Sat, 15 Mar 2025 21:16:37 +0000
Newsgroups: forum.ada-lang.io

This thread is tightly related to [GSoC 2025] Ada's participation? [1]

Google has a program, called Google Summer of Code (GSoC for short) that pays new open source contributors to work on topics for selected projects. Some of these projects could potentially see some work done in Ada. Here is a bit more information for those interested.

- GSoC will accept proposals from contributors (more about what is a contributor later) to any project as long as the mentors/administrators accept it.
- Here is the thing: the administrators of projects and mentors submit a list of ideas for GSoC to fund. BUT, in the end, it is the contributor who has to make a proposal and write it down. This means that even if the administrators do not create a specific project idea/proposal about Ada, if they find the proposal from a contributor interesting, they may accept it. This means that someone could contribute to projects such as GCC/GNAT, RTEMS, etc; as long as there is a serious proposal, there is someone willing to mentor and the administrators accept it.
- What does it mean/take to be a contributor? Well, anybody nowadays can be a contributor, you no longer have to be a student… HOWEVER, there is a catch, not everybody can be one.
- The goal of GSoC is to promote new open source contributors. This means that people with great experience on a topic cannot participate on that topic. AND people who have already contributed to open source projects in a somewhat serious manner, CANNOT be contributors within GSoC. So mainly, it is for young people who are new to the open source world and for professionals who would like to get started in this world. Here is more information Get Started | Google Summer of Code [2]
- There are three types of potential project proposals depending on their time/complexity/size: small (90h), medium (175h) and large (350h). Only projects that could fit in those categories more or less can be carried out.
- And all proposals and potential contributors need to be serious. Students with finals and thesis defenses will most likely not be accepted as they will most likely not have time. Working people with children may not be accepted as they may lack the time. There needs to be a clear commitment and a real possibility of pulling it off!
- Being a mentor takes time and effort! So if you want to mentor someone, take it very seriously! Though you are not a helicopter, the contributor is there to learn.

All the information can be found here [3] (student guide) and here [4] (mentor guide).

From what I know, someone could propose the following projects (but feel free to add your own ideas!):

- Adding parallel keyword support to GCC/GNAT. @sttaft said that he would

be willing to mentor the person. This sounds like a rather complex but fun project. I know Tucker did already some work here in Parasail. Here is GCC's GSoC page [5]

- Upstreaming NetBSD/OpenBSD/other OS support of GNAT to GCC; I may be willing to mentor a person here. This could be a fairly small project, but it requires quite a bit of knowledge! Here is NetBSD's page [6], FreeBSD's [7]
- Improving Ada support in RTEMS. I do not know of any mentor, but maybe someone from the RTEMS team may be willing to help. And these are the kinds of projects that GSoC wants, the ones that help everybody but the main maintainers may not have enough time of willingness to do it themselves. Here is RTEMS's GSoC page [8]
- Adding Ada support to the Zephyr RTOS. I do not know of any mentor here, but an Ada user or potentially a Zephyr maintainer may be willing to help. This is very interesting as Zephyr is a rapidly growing RTOS and we could use that! Here is Zephyr's GSoC page [9]
- The Unikraft microkernel, which is specialised in quick startup environments to run targeted tasks. Here is Unikraft's GSoC page [10]

The deadline for submissions starts on March 24th and ends on April 8th.

Best regards and let's see if someone benefits and steps up!

[1] https://forum.ada-lang.io/t/gsoc-2025-adas-participation/1470

[2] https://summerofcode.withgoogle.com/get-started

[3] https://google.github.io/gsocguides/student/

[4] https://google.github.io/gsocguides/mentor/index

[5] https://summerofcode.withgoogle.com/programs/2025/organizations/gnu-compiler-collection-gcc

[6] https://summerofcode.withgoogle.com/programs/2025/organizations/the-netbsd-foundation

[7] https://summerofcode.withgoogle.com/programs/2025/organizations/the-freebsd-project

[8] https://summerofcode.withgoogle.com/programs/2025/organizations/rtems-project

[9] https://summerofcode.withgoogle.com/programs/2025/organizations/the-linux-foundation

[10] https://summerofcode.withgoogle.com/programs/2025/organizations/unikraft

From: sttaft <sttaft@forum.ada-lang.io>
Date: Thu, 20 Mar 2025 18:39:42 +0000

Hi there, we have put together a team of a contributor and two co-mentors to go after the first project, namely adding parallel keyword support to GCC/GNAT. Stay tuned for more details. So far we are having a positive reaction to our proposal.

From: Micronian2 <Micronian2@forum.ada-lang.io>
Date: Thu, 20 Mar 2025 19:44:55 +0000

By chance do you envision the contributor leveraging the tasking library you had announced last year to aid with the parallel implementation?

From: sttaft <sttaft@forum.ada-lang.io>
Date: Thu, 20 Mar 2025 20:18:20 +0000

Yes, that is exactly right. This project will be connecting the Ada "parallel" syntax to the light-weight-threading library mentioned on this Forum earlier. One nice feature of that library is that it sits "on top" either of a fully standard Ada runtime, or on top of OpenMP, selectable with a simple control-object declaration in the main subprogram or an Ada task body that plans to use light-weight threading.

Here is a slide presentation on the "layered" approach we will be implementing: Ada 202X Lightweight Parallelism and OpenMP.pptx - Google Slides [1]

[1] https://docs.google.com/presentation/d/156vq44aK2FF60cbd_I8hIOXmqEGjl1H7/edit?usp=sharing&ouid=102774392644513331174&rtpof=true&sd=true

From: Irvise <Irvise@forum.ada-lang.io>
Date: Sun, 23 Mar 2025 17:00:17 +0000

Just for completeness, transparency and future reference/traceability, this is the proposal that was sent to the GCC mailing list Proposed GSoC project to enhance the GCC GNAT Ada front end [1] (the rendering is a bit off).

[1] https://gcc.gnu.org/pipermail/gcc/2025-March/245759.html

Ada Mars Rover Demo

From: JeremyGrosser <JeremyGrosser@forum.ada-lang.io>
Subject: Ada Mars Rover demo
Date: Sat, 22 Mar 2025 00:41:18 +0000
Newsgroups: forum.ada-lang.io

AdaCore (@Fabien.C) recently built a demo of what a Mars rover running Ada/SPARK could look like and Anthony Aiello blogged about proving safety properties with it.

GitHub - Fabien-Chouteau/ada-mars-rover [1]

Let's Write a Safety Monitor for a Mars Rover! [2]

by M. Anthony Aiello – Mar 21, 2025. The Ada Mars Rover shouldn't crash into obstacles. See how we formalized this property, discovered an unstated assumption in our remote-control mode that could have led the Rover to crash, removed the assumption, ...

[1] https://github.com/Fabien-Chouteau/ada-mars-rover

[2] https://blog.adacore.com/lets-write-a-safety-monitor-for-a-mars-rover

Distributed Interactive Simulation in Ada?

From: zertovitch <zertovitch@forum.ada-lang.io>
Subject: DIS (Distributed Interactive Simulation) in Ada?
Date: Sat, 22 Mar 2025 17:35:02 +0000
Newsgroups: forum.ada-lang.io

Does anyone know of a usable Ada implementation of the Distributed Interactive Simulation [1] standard (IEEE 1278)?

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

From: JeremyGrosser <JeremyGrosser@forum.ada-lang.io>
Date: Sat, 22 Mar 2025 18:24:22 +0000

I found several posts [1] in the comp.lang.ada archive, including a reference to "ADIS" being posted to an FTP site.

As with most public domain Ada code from that era, it can be found on the Walnut Creek Ada [2] ISO images.

From the June 1995 Disc 1 image, under ada/ajpo/tools/atip/adis/ there's documentation, binaries, and zipped source files.

I've unpacked the archives into a git repository:

GitHub - JeremyGrosser/adis: Ada Distributed Interactive Simulation (IEEE 1278) [3]

[1] https://usenet.ada-lang.io/comp.lang.ada/?q=distributed+interactive+simulation&r

[2] https://archive.org/search?query=walnut+creek+ada

[3] https://github.com/JeremyGrosser/adis

From: cup <cup@forum.ada-lang.io>
Date: Sun, 23 Mar 2025 20:06:18 +0000

I wonder how complete this implementation is. Most of the implementations I've used are in C or Java. [...]

Ada-related Tools

Alire on Native ARM Hosts?

From: pchapin <pchapin@forum.ada-lang.io>
Subject: Alire on native ARM hosts?
Date: Fri, 3 Jan 2025 13:38:49 +0000
Newsgroups: forum.ada-lang.io

I'm hoping to contribute in a useful way to the Ada open source community once I retire at the end of this year. To that end, I've been experimenting with Alire.

I notice that on my M-series MacBook, the "native" version of GNAT that Alire offers is actually x86_64, and generates x86_64 executables. This works because of Rosetta (Apple's binary translation system), but it's not ideal. The use of x86_64 on my ARM MacBook is a suitable transitory practice, but I would have expected that by now Alire's concept of "native" would be an ARM compiler for my host.

I ran into the same problem on my new Windows-on-ARM laptop, but it's more to be expected there since Windows-on-ARM is less common.

Is there any timeline on when a real native (meaning ARM generating) compiler will be available for macOS or Windows?

As an aside, I notice that the Rust community has a true ARM-native compiler for macOS /and/ Windows. Just sayin'.

From: Irvise <Irvise@forum.ada-lang.io>
Date: Fri, 3 Jan 2025 13:47:02 +0000

There is already a native build of GNAT for Apple silicon devices thanks to @simonjwright [1].

The reason it is not available by default in Alire is that Alire builds everything using Github's runners. As there is currently no availability for ARM in Github (though they already announced they plan on letting runners do tasks on ARM OSX), there is no built-in Alire support for Apple-ARM. But you can already use it using the link above and with time it shall become available for everybody else.

Thanks for wanting to help the wider Ada community ^^ Welcome to the forums and good luck with your retirement!

[1] https://github.com/simonjwright/building-gcc-macos-native

From: AJ-Ianozi <AJ-Ianozi@forum.ada-lang.io>
Date: Fri, 3 Jan 2025 13:57:40 +0000

For MacOS, we do have the native build. If you install Alire via https://getada.dev [1] on an apple silicone Macbook it'll automatically install the aarch64 version. Likewise the "Download Alire" button on https://ada-lang.io/ [2] defaults to aarch64 MacOS.

On Windows it's a little more complicated since it's going to need MSYS2 for arm64, which it looks like there's some experimental builds (of MSYS2, not Alire): [3]

The biggest hurdle to having other platforms besides x64 and aarch64 on mac is that everything (including Alire's binary releases) is built via Github actions, and github only has aarch64 for macOS runners right now. However, that should be changing this year, and I suspect we'll have at least arm64 builds for GNAT in 2025.

[1] https://getada.dev

[2] https://ada-lang.io/

[3] https://github.com/Windows-on-ARM-Experiments/msys2-woarm64-build

From: pchapin <pchapin@forum.ada-lang.io>
Date: Fri, 3 Jan 2025 14:15:00 +0000

Thanks for your reply, but I have two questions.

First, I'm not as concerned about Alire itself. While it would be nice for Alire to be ARM64, I'm more concerned about the architecture of the code produced by the compiler Alire is giving me. Are you saying that the ARM64 version of Alire will also fetch an ARM64 generating version of the compiler?

Second, and this is more an observation than a question, but it seems weird to me that what appears to be the official site for Alire (https://alire.ada.dev/) is only offering an x86_64 version of the program to macOS users when an ARM64 version exists elsewhere. As someone new to Alire, it feels like there might be a lack of coordination within the community.

From: jere <jere@forum.ada-lang.io>
Date: Fri, 3 Jan 2025 18:29:30 +0000

> I'm more concerned about the architecture of the code produced by the compiler Alire is giving me

Alire is mostly platform independent. It uses the compiler you tell it to. So if you tell it to use an ARM64 compiler and one is available it will use it. It can use compilers from its own pool or ones you supply, all at your choice. Hopefully that helps clarify it a bit more?

> it feels like there might be a lack of coordination within the community.

I would definitely encourage you to reread the reply from @Irvise as it explains this. This isn't a case of coordination issues. This is a case of what technology supports and doesn't. Github doesn't support that architecture for its runners (right now. It will in the future hopefully). Alire relies on Github's runners, so it cannot handle that architecture built in yet. So as not to leave people that need that architecture out, Alire allows you the user to add compilers that it cannot provide itself (due to lack of support on Github). Hopefully that clarifies it more, but please reread his post.

From: pchapin <pchapin@forum.ada-lang.io>
Date: Fri, 3 Jan 2025 22:10:20 +0000

I understand that there are some technical issues with GitHub. However, an ARM64 version of Alire and GNAT do exist via ada-lang.io [1]. Why can't alire.ada.dev link to those resources so everything appears in one place, or is at least consistent between sites?

Someone new to Ada is likely to go to alire.ada.dev to get Alire. There they only see x86_64 tools. Are they going to persevere enough to find a different site where ARM64 tools exist? I'm going to guess many won't.

In any case, this situation makes the Ada community look fractured. Is there some reason why the people who run alire.ada.dev won't link to the ARM64 version of Alire on ada-lang.io? Do the people who run those respective sites talk to each other? Is this symptomatic of some kind of internal struggle in the Ada community? Or perhaps the version of Alire on ada-lang.io is substandard in some way. These are the kind of questions I would ask as a newcomer to the community.

From: pyj <pyj@forum.ada-lang.io>
Date: Sat, 4 Jan 2025 03:22:40 +0000

ada-lang.io [1] doesn't have any of the actual Alire binaries, it just points to artifacts produced by the Alire team on GitHub.

> Do the people who run those respective sites talk to each other

AFAIK, there's been no coordination between the two. There's a monthly community Ada meeting run by @Irvise, but I don't know who attends.

I originally started ada-lang.io as an open source community site, but handed it off to Jeremy who runs it due to several important family matters at the time. It's public and anyone can send a pull request to update it.

> Is this symptomatic of some kind of internal struggle in the Ada community?

Not that I'm aware of. I took some flak for "speaking for the community" from some people with the site, so it probably could be branded better.

Other than learn.adacore.com, the onboarding ramp for people into Ada,

especially for cross-platform non-embedded development is much worse than other languages.

From: pchapin <pchapin@forum.ada-lang.io>
Date: Sat, 4 Jan 2025 14:05:00 +0000

> ada-lang.io [...] just points to artifacts produced by the Alire team

That begs the question as to why doesn't alire.ada.dev link to the ARM64 version for macOS. It seems like alire.ada.dev isn't being updated to reflect current reality, and that ada-lang.io [1] is a more appropriate place for newcomers to get Alire… something that would not be obvious, particularly given that Alire's documentation is on alire.ada.dev.

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Sat, 4 Jan 2025 10:52:21 +0000

The problem for the websites (https://ada-lang.io and https://alire.ada.dev) is that Apple goes to considerable lengths to make it impossible hard for the server to detect the client architecture. The User-Agent string tells you it's a Mac and whether it's Safari of Firefox (or, I guess, Chrome, or …) but the rest doesn't distinguish.

ada-lang.io decided to offer an aarch64 binary for alr; alire.ada.dev still offers the x86_64 binary.

Github has provided aarch64 runners for some time now; there are some limitations, e.g. macos-14 is aarch64, earlier versions are x86_64. There are also Apple's SDK vagaries. But for the moment things are stable.

If you look at the ALR Release pages (2.0.2 [1] and Nightly [2]) you'll see that all the parallel releases are at the same date, strongly implying that they were built at the same time in the same environment (Github).

There are at least 3 ways ahead from here:

- allow the server to find the client architecture, so the right binary can be downloaded,
- provide a universal binary (this requires merging already-built aarch64 and x86_64 binaries, so a change to the build process),
- fix alr so it decides which compiler version to download based on the runtime architecture rather than the architecture on which it was built.

[1] https://github.com/alire-project/alire/releases/tag/v2.0.2

[2] https://github.com/alire-project/alire/releases/tag/nightly

From: pchapin <pchapin@forum.ada-lang.io>
Date: Sat, 4 Jan 2025 14:10:23 +0000

> There are at least 3 ways ahead from here:

An easier thing would be to provide multiple links: one for the x86_64 version and one for the ARM64 version. Let the user decide which one to download. Given that Alire is targeting developers, it seems reasonable to assume they would know the architecture they want. Also, this approach is common on other websites I've seen.

As an aside, I tried installing the x64 version of Alire on my new Windows-on-ARM laptop. It didn't work at all. Specifically, there is some issue with the handling of the console that causes Alire to print out a large number of blank lines instead of normal text.

I get that it might be a while before there is a native WoA version of Alire, but I just wanted to throw my observations out there for general information.

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Sat, 4 Jan 2025 14:14:12 +0000

> fix alr so it decides which compiler version to download based on the runtime architecture rather than the architecture on which it was built.

The trouble with this is that it would require alr to be x86_64, so that it would work equally on either architecture, which would mean running under Rosetta2 on aarch64, which might confuse users.

(Also, I just spent too long trying to work out how to do this: not helped by the fact that my Ada Language Server setup doesn't work at all well with the Alire code structure).

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Fri, 10 Jan 2025 17:16:51 +0000

After considerable trouble (note to self: start your development from the latest upstream branch!) I've built a macOS universal binary of alr, using the latest (nightly) release; see here https://github.com/simonjwright/alire/releases/tag/nightly

From: kevlar700 <kevlar700@forum.ada-lang.io>
Date: Sat, 4 Jan 2025 11:46:16 +0000

> makes the Ada community look fractured

A significant issue is that there should really be one domain for important tooling a user is expected to run. Go actually had a different site before consolidating on go.dev. For ages users kept going to the original. One domain means a user can look and say okay I shall trust any downloads from this domain. Currently I think crates are approved manually but github name space attacks that Go made efforts to deal with should also be kept in mind if Ada attracts the growth that it deserves.

From: pchapin <pchapin@forum.ada-lang.io>
Date: Fri, 3 Jan 2025 22:01:16 +0000

Great! I can confirm that works. I downloaded the ARM64 version of Alire from ada-lang.io, selected the "native" toolchain, and got a compiler that outputs ARM64 executables. That's wonderful, thanks!

To be clear, this was on my MacBook. I understand that ARM64 support for Windows isn't really a thing yet.

Simple Components 4.70

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components 4.70
Date: Sat, 18 Jan 2025 15:26:19 +0000
Newsgroups: comp.lang.ada

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), 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

Changes to the previous version:

- The package Unbounded_Unsigneds implementing arbitrary precision unsigned arithmetic was added;
- The package Unbounded_Integers implementing arbitrary precision integer arithmetic was added;
- The package Unbounded_Unsigneds Primes implementing operations with prime numbers was added;
- The package Unbounded_Unsigneds.Montgomery implementing Montgomery domain operations was added;
- The package Unbounded_Unsigneds.Barrett implementing Barrett reduction was added;
- The package Strings_Edit.Unbounded_Unsigned_Edit string editing for arbitrary precision unsigned numbers was added;
- The package Strings_Edit.Unbounded_Integer_Edit string editing for arbitrary precision integer numbers was added;

- The package Unbounded_Unsigneds.Parallel implementing parallel arbitrary precision algorithms was added;
- The package Job_Servers was added implementing servers of jobs backed by a task pool;
- The number protocol added to Python bindings;
- Reply_Text and Reply_HTML in GNAT.Sockets.Connection_State_Machine.HTTP_Server modified to call Send_Body passing the Get parameter rather than skipping it when Get = False;
- UUID v6 and v7 generation was added to Universally_Unique_Identifiers.

The key points of the arbitrary precision arithmetic packages design and functionality:

- Advanced memory management preventing excessive copying;
- The number internal representation vector is shared between objects if possible;
- No limit on the number size, except for the storage pool size;
- In-place versions of operations (e.g. for addition, subtraction) further reduce need of copying;
- Lazy memory deallocation strategy, the memory is kept between variable updates;
- Swapping variables;
- Long to short operations;
- Squaring;
- Square root, square root with remainder;
- Multiplicative inverse;
- 2's complement;
- Bit representation access, slicing, truncation;
- Full division with remainder, remainder only division;
- Karatsuba multiplication and squaring;
- Specialized operations involving powers of two and words;
- Exponentiation under modulo;
- Fibonacci number under modulo;
- Montgomery domain multiplication, squaring, exponentiation under modulo and primality tests of the domain modulus;
- Barrett reduction, multiplication, exponentiation;
- Primality tests: Miller-Rabin, Fibonacci, Lucas-Lehmer, strong Lucas;
- Parallel algorithms for very large numbers;
- String editing and formatting packages for the numbers.

Performance notes. In order to get optimal performance -O2 switch need to be used. It does 3x performance boost. 64-bit (with 128-bit integer) outperform 32-bit by many multiplies. See GPR variables: https://www.dmitry-kazakov.de/ada/components.htm#19 E.g.

gprbuild -P components-tests.gpr -XTarget_OS=Linux -Xarch=aarch64 -XDevelopment=Release

Simple Components v4.71

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components v4.71
Date: Thu, 30 Jan 2025 15:10:14 +0000
Newsgroups: comp.lang.ada

[Repeated information omitted. —arm]

Changes to the previous version:

- The package Generic_FFT provides an implementation fast Fourier transform. The implementation supports pre-computed bit-reverse permutation and exponents to be used in multiple transformations of same vector length.

Simple Components v4.72

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components v4.72
Date: Tue, 11 Mar 2025 17:23:13 +0000
Newsgroups: comp.lang.ada

[Repeated information omitted. —arm]

Changes the previous version:

- Topic list bug fix in the package GNAT.Sockets.MQTT (thanks to Xavier Grave);
- Minor performance improvements in Unbounded_Unsigneds: calculating log2, testing for power of two;
- OpenSSL MQTT test (components-connections_server.mqtt-test_mqtt) added;
- GNAT.Sockets.Server.Secure (TLS) bug fixed. Activated is called at the end of handshaking;
- GNAT.Sockets.Server.OpenSSL bug fixed. Activated is called at the end of handshaking.

Simple Components v4.73

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components v4.73
Date: Sat, 29 Mar 2025 11:35:06 +0000
Newsgroups: comp.lang.ada

[Repeated information omitted. —arm]

Changes the previous version:

- Bug fix in Generic_Set.Insert which failed to set Inserted output correctly when the element is already in the set;
- Bug fix in Generic_Indefinite_Set.Insert, same as above;
- The package Generic_Lock_Free_FIFO was added. The package implements lock-free N-to-M FIFOs.

Unit Testing Framework Recommendation

From: mosteo <mosteo@forum.ada-lang.io>
Subject: Unit testing framework recommendation
Date: Tue, 21 Jan 2025 14:58:05 +0000
Newsgroups: forum.ada-lang.io

I know of the existence of aunit, ahven, gnattest, but my experience is anecdotal and outdated. Before doing my own research, maybe some of you already have the answer ready.

I'm looking for something that generates the boilerplate (harness project? and test stubs) and is able to find new subprograms in subsequent generation runs without botching existing tests.

Ada for the tooling is not mandatory, as long as the generated test stubs are in Ada. Not sure if such a beast exists though…

From: pyj <pyj@forum.ada-lang.io>
Date: Wed, 22 Jan 2025 01:48:38 +0000

I'm not aware of any.

Trendy Test doesn't use any generated code, but instead hijacks the exception system to handle registration and running, relying on GNAT unused procedure warnings to ensure that all tests get added to the test pools.

The most likely way to get auto-registration to work as-is would be to write a test framework that follows what Criterion [1] does. It uses macros to write test functions to a special separate linker section and then read that out of the ELF or COFF at test startup time to effectively give it the ability to dynamically determine the list of all tests to run. This relies on having appropriate binary analysis to look at either of these file types at startup.

[1] https://github.com/Snaipe/Criterion

From: pyj <pyj@forum.ada-lang.io>
Date: Wed, 22 Jan 2025 13:59:41 +0000

Thinking about this some more, you wouldn't even have to parse the elf/coff, if you made a custom pragma for unit test registration (similar to #[test] for Rust, iirc, C# and F# have something similar), the implementation should ignore it, so you could write a script to extract them, and dump them into an entry procedure (with some wrapper for catching exceptions and reporting). I had thought of pitching an RFC a while ago for a Unit_Test attribute, but thought it might be too specialized.

```
pragma Unit_Test (Test_Function);
-- Maybe it should be a function instead, idk.
procedure Test_Function is begin ... end;
```

Something like:

```
find . name '*.ads' -o -name '*.adb' | egrep
'pragma\s+Unit_Test \((.*)\) |
write_test_wrapper.sh > run_tests.adb'
```

I've wanted to update trendy test anyways to add property testing and some other stuff, I'll try this tonight.

From: mosteo <mosteo@forum.ada-lang.io>
Date: Thu, 23 Jan 2025 10:28:19 +0000

> if you made a custom pragma for unit test registration

Frankly, before I reach the point of registering custom tests, I'd like to have automatically registered tests for every visible subprogram that XFAIL until I fill them.

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Thu, 23 Jan 2025 17:01:09 +0000

I tentatively push my Scripted Testing [1] crate - it's intended for functional tests rather than unit tests.

[1] https://github.com/simonjwright/scripted_testing

From: ebriot <ebriot@forum.ada-lang.io>
Date: Thu, 23 Jan 2025 18:07:55 +0000

Just curious: in what domains are people actually using extensive unit tests (as opposed to functional tests or integration tests that test whole packages or projects)? I have started a few times with the intention of having unit tests for a library or something, but it generally becomes quite visible that this is a major test that's not worth the trouble in general. Also it somewhat deters from refactoring the code and splitting it into more subprograms.

From: jere <jere@forum.ada-lang.io>
Date: Thu, 23 Jan 2025 19:46:44 +0000

For me it depends on what is being developed. I work on a lot of custom message protocols, so when we have a brand new one, I do heavy unit testing on them to ensure there are no bugs in the implementation (and to some extent catch any conflicts in the design). If it's a more vetted system, I'll rely only on functional testing.

On my hobby projects: The NES emulator that I am putzing around with has a ton of unit tests for the CPU. I wanted to make darn sure my instruction execution was as sound as possible as I figured it was the hardest place to debug later on once I integrated the other parts. Now that I am past that part I'll probably rely more on functional testing.

I find a good balance works well for me. I don't know of any good tools for Mosteo though. I've always rolled my own, which is time consuming.

EDIT: I guess if someone was really feeling froggy they could make an application based on libadalang that just parsed files and looked for library level function / procedure declarations and generated the stubs. Might be a fun project for someone.

From: pyj <pyj@forum.ada-lang.io>
Date: Thu, 23 Jan 2025 21:56:05 +0000

> they could make an application based on libadalang

I don't know about libadalang, but I could probably do it in Raffle. I was planning on doing this to be able to automatically make Lua bindings for packages, and also to make better searchable docs based on spec files, like "I have a Foo, what subprograms return a Foo and which take a Foo as a parameter?"

> Frankly, before I reach the point of registering custom tests, I'd like to have automatically registered tests for every visible subprogram that XFAIL until I fill them.

Oh, I do unit tests differently where I focus tests on cases, not necessarily specific subprograms. I see what you mean now.

> in what domains are people actually using extensive unit tests

My experience has also been that full end-to-end tests are much more powerful and long-lived. BBT [1] has been good for this. Unit tests are great for "did I write this thing correctly?" but I also very liberally use Pre/Pre and constraints and run with all checks on.

[1] https://github.com/LionelDraghi/bbt

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Thu, 23 Jan 2025 22:02:09 +0000

My Scripted Testing crate uses libadalang (well, libadalang2xml) and generates code using xslt. See the demo/'s README [1].

Unfortunately I think it may be the opposite to what you want - it replaces the bodies of a called package with scripting support.

[1] https://github.com/simonjwright/scripted_testing/tree/alire/demo

From: Fabien.C <Fabien.C@forum.ada-lang.io>
Date: Fri, 24 Jan 2025 16:07:26 +0000

> they could make an application based on libadalang [...] and generate the stubs

This is exactly what GNATtest [1] is ^^

[1] https://github.com/AdaCore/libadalang-tools

From: mosteo <mosteo@forum.ada-lang.io>
Date: Thu, 23 Jan 2025 22:08:40 +0000

> in what domains are people actually using extensive unit tests

In my case it is small focused libraries with relatively few public subprograms. I was thinking that it would be better to start with these tests rather than with an ad-hoc, haphazard collection of tests (which is what I usually end up doing).

But it may well turn out that what I want to try is not really practical.

From: LionelDraghi <LionelDraghi@forum.ada-lang.io>
Date: Fri, 24 Jan 2025 18:02:55 +0000

My three *small* open source apps (acc, smk and bbt) are CLI oriented, and easy to test end-to-end.

I have very very rare needs for unit testing. Last example is a function to compute a relative shortest path from, lets say, /home/lionel/bin/x to /home/lionel/y/z : taking into account the Windows drive, Windows vs Unix separators, etc.

This easily leads to many test cases.

So there is one unit test, not even for a package, just for this procedure.

I noticed that in CLI, there are often commands or options that do not execute the full process the app is intended to run, and activate only a small part of the code. For example dry-run options, or command to list input files recursively found by the app.

Those commands or options sometimes offer a way to have a stable end-to-end test on a limited scope, that would have been an integration or a unit test otherwise.

Going further, I don't exclude if the need arises to cheat, and have a secret option exposing some *stable* internal data only for test purposes.

Using those internal data in (false) end-to-end tests, depending on the command line interface and not on the package/procedure profiles, seems to me less prone to change when the code evolves, and does not deter refactoring.

All in all, I have 98% end-to-end tests, 2% unit testing, no integration tests.

My small CLI utilities seem to me an easy case for testing, things would probably be not so simple on a larger app, server, app with GUI, app with high coverage requirement, etc.

From: zertovitch <zertovitch@forum.ada-lang.io>
Date: Wed, 22 Jan 2025 21:59:43 +0000

> I'm looking for something that generates the boilerplate [...]

This code generation would be a perfect job for HAC [1].

[1] https://hacadacompiler.sourceforge.io/

From: Fabien.C <Fabien.C@forum.ada-lang.io>
Date: Fri, 24 Jan 2025 16:13:24 +0000

Cesar who joined my team a couple months ago will experiment with a simple test framework built-in Alire.

Along the lines of:

- Find all the files matching a given pattern in a directory (e.g. tests/). Each file is an ada “Main” procedure .
- Generate a GPR file to compile all these files as executables (one exec for each)
- Run them one by one and gather results

Writing a test would be as simple a writing a single source file:

```
with My_Lib;
procedure Test is
begin
  pragma Assert (My_Lib.Double (1) = 2);
end Test;
```

From: ebriot <ebriot@forum.ada-lang.io>
Date: Sat, 25 Jan 2025 16:27:55 +0000

We have an asserts package (a generic of course), based on GNATCOLL.Asserts. Much nicer to use than a straight pragma Assert, because in case of errors you see both sides of the “=” (or any other operator). Also has support for retrying (for those times when you are dealing with asynchronous things), comparing arrays,… I am sure a lot of people have similar packages, or maybe use GNATCOLL.Asserts directly… @Fabien.C maybe Cesar could also gather ideas and later improve that package to make writing tests even nicer.

From: krischik <krischik@forum.ada-lang.io>
Date: Sun, 26 Jan 2025 18:09:32 +0000

I just use AUnit and just write my own. Yes, AUnit needs a lot of boilerplate but it’s mostly copy paste and I wrote my own extensions to make it more bearable.

> if you made a custom pragma for unit test registration

That would indeed be nice. There is a reason all other unit test frameworks I know of do it that way.

> in what domains are people actually using extensive unit tests (as opposed to functional tests or integration tests that test whole packages or projects)

It do. It’s called test driven development where you write the test first and then the method. But more importantly: It’s not either or. You can do both.

Which is what I suggest for Android. With the unit test designed to run without the actual device. Since testing on device is significantly slower.

> I also very liberally use Pre/Pre and constraints and run with all checks on.

I too have the contracts active when running unit tests and it did indeed find problems I would not have found otherwise.

> Find all the files matching a given pattern in a directory (e.g. tests/). Each file is an ada “Main” procedure

I hope that won’t break my AUnit tests I already have in my Alire crates and already run by the Alire build server.

> I am sure a lot of people have similar packages

Indeed I have: Fluent AUnit Asserts · Ada Class Library [1]

[1] https://adacl.sourceforge.net/adacl-assert/

From: Trescott <Trescott@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 07:00:49 +0000

> That would indeed be nice. There is a reason all other unit test frameworks I know of do it that way.

@krischik I may be misreading this, but I think you might have missed something here. In case you are unaware, I’ll point this out, but maybe you do already know. (I mean no offense.)

Ada compilers are permitted to ignore pragma statements that are not understood by the compiler. (And most do, I think, including GNAT) See ARM 2.8, paragraph 15 [1] So, with an existing compiler I can add made-up pragma statements to my code, and it will still compile. (But I will get warnings that the pragma was not understood and was ignored.)

Try adding your own made up pragma statement to some code and see for yourself. Such as:

pragma Quack_Like_A_Duck; --No compiler "supports" this pragma, but it will still compile.

So, @pyj is not suggesting adding support in the compiler, but rather taking advantage of the existing permission to ignore unknown pragmas. Semantically correct Ada files, that compile successfully, can be marked up using a made-up pragma. Then some tool can search for those made-up pragmas.

[1] https://ada-lang.io/docs/arm/AA-2/AA-2.8#p14

From: csagaert <csagaert@forum.ada-lang.io>
Date: Mon, 3 Feb 2025 14:00:58 +0000

I posted a first draft [1] of what I had in mind for a built-in testing feature, on the Alire github repository. This first draft is intended to collect ideas and feedback, in order to build something that would be a good fit for actual crates in the wild so feel free to comment and add ideas!

[1] https://github.com/alire-project/alire/issues/1831

Linux Distribution for Alire

From: alexispaez <alexispaez@forum.ada-lang.io>
Subject: Linux distribution for alire
Date: Sat, 8 Feb 2025 19:36:39 +0000
Newsgroups: forum.ada-lang.io

I’ve been using Alire on Windows for some time now, without too many quirks coming up. I’ve also tested it on a very old laptop that is running Kubuntu 24.04 to find it is surprisingly quick on it. I’d now like to try it out on WSL on my Windows 11 PC. I've been using WSL at work and it seems to work quite well.

Since I’m able to choose, what Linux distro would you recommend for working with Alire?

From: mosteo <mosteo@forum.ada-lang.io>
Date: Wed, 12 Feb 2025 10:38:12 +0000

The biggest difference will be on supported system packages. If you go for one of our unsupported or less used distros, you’ll have no (or fewer) system packages immediately available, if you need those.

The distros with more supported system packages will be, I expect, in decreasing order: Ubuntu, Debian, Arch.

You could also check here https://alire-crate-ci.ada.dev/pages/status.html if some crates of special relevance to you are available in the distro you prefer.

SPARK for MacOS on Apple Silicon?

From: Ret_Build_Engineer <Ret_Build_Engineer@forum.ada-lang.io>
Subject: SPARK compiler (etc) for MacOS on Apple Silicon? I don't see gnatprove etc
Date: Sun, 9 Feb 2025 19:59:43 +0000
Newsgroups: forum.ada-lang.io

Pardon me in advance if this is a dumb/obvious question…

SPARK compiler (etc) for MacOS on Apple Silicon? I don’t see gnatprove etc

I was looking at some of the Rosetta Code examples written in SPARK and I thought I might start giving SPARK some attention.

So I’m going through the SPARK section of AdaCore Learn web site.

Perhaps this can only be done through Alire?

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Sun, 9 Feb 2025 22:29:42 +0000

You can download it using alr install gnatprove. In my case it can then be found in ~/.alire/bin. Assuming you're using an aarch64 alr it'll be the aarch64 version - but even if you aren't it'll almost certainly run under Rosetta2.

Installing GNAT LLVM

From: gcalliet <gcalliet@forum.ada-lang.io>
Subject: Installing gnat llvm
Date: Tue, 11 Feb 2025 15:47:27 +0000
Newsgroups: forum.ada-lang.io

Some troubles installing gnat llvm

I'm on Ubuntu 24

Installed llvm-16, clang16

Doing what is said on the readme

Error: command opt not found

If I install llvm-dev llvm 18 is installed and opt is found

gprbuild : c++ compiler not found

I install clag , clang 18 installed

Now:

compile llvm_wrapper.cc
llvm/Support/AArch64TargetParser.h file not found

Where do I have to search?

From: Irvise <Irvise@forum.ada-lang.io>
Date: Tue, 11 Feb 2025 22:54:04 +0000

First things first, welcome to the forum!

And you are quite lucky! I just compiled GNAT-LLVM for the first time myself over the weekend, so I think I can help you.

GNAT-LLVM currently requires LLVM version 16. No other version is expected to work (LLVM has breaking changes between major releases). You can use the package manager binaries. That is recommended as it is the quickest and simplest way to get started. You will need both llvm-16, clang16 and their -dev versions. The -dev versions also have to match the version of llvm/clang, so it needs to be 16.

You will also need to have gnat/gcc installed as well as GPRBuild, which seems to be your case.

Once you have everything correctly set up, you can proceed as the README says, which it seems that you have already done. However, as you are getting a compilation error and it seems related to LLVM sources, this may be related to you having version 18 also installed, which may be the one being selected as the base LLVM.

In order to solve this issue, the GNAT-LLVM repo has a failback mechanism: to compile a local copy of LLVM-16 directly as part of the GNAT-LLVM build process. This is time consuming and requires a somewhat powerful machine or it will take a while. To do this, follow the README instruction: run make llvm on the root folder and it will build the local copy. Afterwards, you can run a simple make and it should work.

If you have enough RAM and quite a few cores, I would suggest using parallel compilation of LLVM by using make llvm -j and then just issue a simple make (do not add the -j flag to the normal make, as it already uses it internally).

If you get further issues, please, let me know. Also, if your GCC/GNAT compiler is version 14, you will need to use the patches indicated here [1]

Best regards and happy Ada hacking!

[1] https://github.com/godunko/adawebpack/tree/master/patches

Gnathtml.pl Replacement

From: zertovitch <zertovitch@forum.ada-lang.io>
Subject: Gnathtml.pl replacement
Date: Tue, 11 Feb 2025 21:01:10 +0000
Newsgroups: forum.ada-lang.io

Did anyone try to replace gnahtml.pl with something else, possibly based on libadalang?

gnahtml.pl is a Perl script from AdaCore that produces a set of Web pages from Ada sources, by scanning the .ali files produced by the GNAT compilation.

It works fine but is a bit difficult to customize. Also it is based on deprecated HTML frames (although browsers support them).

From: bracke <bracke@forum.ada-lang.io>
Date: Wed, 12 Feb 2025 07:00:42 +0000

Yes, there is:
https://docs.adacore.com/live/wave/gnatdoc4/html/gnatdoc-doc/introduction.htmll

From: zertovitch <zertovitch@forum.ada-lang.io>
Date: Thu, 13 Feb 2025 18:39:12 +0000

Lovely!

Strange, the bodies' sources are not shown under "Source Files", despite the -b, -d and -p options.

Is there a way to make them appear?

Actually I am looking for a simple source browser - basically like what is generated by gnathtml but with "modern"-looking HTML frames.

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Thu, 13 Feb 2025 19:58:42 +0000

My copy of gnatdoc=25.0.0 doesn't have -b, -d, -p options. The (next-generation) documentation [1] suggests you should use --generate=body. But when I try gnatdoc it fails with

raised ADA.IO_EXCEPTIONS.NAME_ERROR : Could not open gnatdoc/share/gnatdoc/html/template/index.xhtml

in spite of there being such a file in ~/.alire/share/gnatdoc/html/template/index.xhtml.

[1] https://docs.adacore.com/live/wave/gnatdoc4/html/gnatdoc-doc/introduction.html

From: zertovitch <zertovitch@forum.ada-lang.io>
Date: Thu, 13 Feb 2025 20:26:20 +0000

Interesting… The 23 (Pro) version shows, among others:

-b Process bodies to complete the spec documentation

-d Document bodies

-p Process private part of packages

Perhaps some options were lost on the road?

Anyway, the old Perl script (gnathtml.pl), reads the .ali files produced by GNAT on compilation and works fine.

Ideally an Ada-Perl bilingual could translate it to Ada. It would be easy to update and customize it then.

From: ebriot <ebriot@forum.ada-lang.io>
Date: Fri, 14 Feb 2025 07:37:38 +0000

I originally implemented gnathtml (in 1998!). Parsing ALI files is simple, but this is actually way more complicated than it looks. Since I first developed the tool, xref in ALI were added for generics, instances, parent types of tagged types, … resulting in quite a complex system of annotations.

I have tried since then to do something similar (in python) to find unused entities in our codebase, but this is at best inaccurate. Maybe I just did not spend enough time on it, but at this point it seems to me that ALI-based is not the way to go if you want an accurate tool

From: ebriot <ebriot@forum.ada-lang.io>
Date: Fri, 14 Feb 2025 19:22:27 +0000

Following up on private messages: for those interested in understanding the ALI xref format, this is documented in lib-xref.ads if the GNAT sources

From: zertovitch <zertovitch@forum.ada-lang.io>
Date: Mon, 17 Feb 2025 19:51:04 +0000

Here is the very beginning of such a tool: [1]

Actually the parsing is quite easy (if you skip some details).

[1] https://github.com/zertovitch/ali_parse

Using Alire on WSL

From: alexispaez <alexispaez@forum.ada-lang.io>
Subject: Using Alire on WSL
Date: Sun, 16 Feb 2025 13:38:41 +0000
Newsgroups: forum.ada-lang.io

Just a heads up on using Alire on WSL2, from what I'm seeing it's faster than using Alire native on Windows (well, it's not really native). The Linux version of GNAT Studio also works. Here's a couple of alr build tests:

- On WSL2 Ubuntu-22.04:
  Hello: 0.53 seconds.
  Aws crate 49.72 seconds.
- On Native Windows 11:
  Hello: 0.77 seconds.
  Aws crate: 70.14 seconds.

I'd say not bad!

Aarch64 GNAT Cross Toolchain for MacOS?

From: jgrivera67 <jgrivera67@forum.ada-lang.io>
Subject: Is there an aarch64 gnat cross toolchain available for MacOS?
Date: Mon, 17 Feb 2025 01:03:39 +0000
Newsgroups: forum.ada-lang.io

Can someone please tell me if there is an ARM aarch64 gnat cross toolchain that I can use with Alire on macOS, and how to get it?

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Mon, 17 Feb 2025 09:42:10 +0000

If you go to the alire-project's Nightly release page [1] you can download the aarch64 [2] or the universal [3] build.

Or you can go to the 2.0.2 release page [4] where you'll find the aarch64 [5] build.

Or you can go to the ada-lang.io home page, where the blue /Download Alire 2.0.2 for macOS/button will download the aarch64 build noted above.

NB! you should run

% xattr -d com.apple.quarantine ~/Downloads/alr-2.0.2-bin/*/macos.zip

before unpacking the download.

These versions of alr will install native aarch64 compilers via alr toolchain --select.

[1] https://github.com/alire-project/alire/releases/tag/nightly

[2] https://github.com/alire-project/alire/releases/download/nightly/alr-nightly-bin-aarch64-macos.zip

[3] https://github.com/alire-project/alire/releases/download/nightly/alr-nightly-bin-universal-macos.zip

[4] https://github.com/alire-project/alire/releases/tag/v2.0.2

[5] https://github.com/alire-project/alire/releases/download/v2.0.2/alr-2.0.2-bin-aarch64-macos.zip

License of a Project

From: EzeMath <EzeMath@forum.ada-lang.io>
Subject: License of a Project
Date: Tue, 18 Feb 2025 12:57:09 +0000
Newsgroups: forum.ada-lang.io

Greetings, I would like to know if using the standard ada libraries that have LGPL or GPLV3 would make my entire code have to be open source.

Someone tell me if this is true

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Tue, 18 Feb 2025 14:06:53 +0000

No.

GNAT's library code comes with this exception:

-- As a special exception under Section 7 -
-- of GPL version 3, you are granted
-- additional permissions described in the
-- GCC Runtime Library Exception,
-- version 3.1, as published by the Free
-- Software Foundation.

Here's the GCC Runtime Library Exception: https://www.gnu.org/licenses/gcc-exception-3.1.en.html

From: EzeMath <EzeMath@forum.ada-lang.io>
Date: Tue, 18 Feb 2025 17:54:41 +0000

One reason for all my code to be covered by the GPL would be that I want to distribute the software and I imagine that if the code remains within the company then it should not release the code.

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Tue, 18 Feb 2025 19:49:02 +0000

Obviously if you develop code on the company's time your contract of employment will/should address the right you have in that code. You'd probably have little personal right, since you're working for hire.

What you do on your own time should be a different question, but sometimes it isn't. At one point my then employer wanted to say that they would have rights in any code I developed. AFAICR I crossed out & initialled those sections in the contract; never heard anything further.

From: jere <jere@forum.ada-lang.io>
Date: Tue, 18 Feb 2025 15:02:39 +0000

If you are using the FSF version of GNAT (no cost, Alire/GetAda gets it for you pretty easily) or GNAT PRO (costs $$$), then you are safe and your own code does not have to be GPL. If you are trying to use an old GNAT Community release then you have a problem (this is less common but I see people still trying to download it from the AdaCore website).

Safe alternatives:

GetAda: https://www.getada.dev/

Alire: https://alire.ada.dev/

Your platform's package manager (pacman, apt get, etc.)

NOTE: GetAda is a wrapper for more easily setting up Alire. Think GetAda=Rustup and Alire=Cargo (not an exact match, but the general idea).

Ada PDF Writer v7

From: zertovitch <zertovitch@forum.ada-lang.io>
Subject: Ada PDF Writer version 7
Date: Sun, 23 Feb 2025 08:31:48 +0000
Newsgroups: forum.ada-lang.io

PDF_Out is an Ada package for producing easily and automatically PDF files, with text, vector graphics, and imported images (JPEG).

With PDF_Out, you can automatically produce reports, invoices, tickets, labels, delivery notes, charts, maps etc. from your Ada program.

The Ada PDF Writer is free and open-source, fully programmed in Ada.

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

Sources, site #1: SourceForge.net [1]

Sources, site #2: GitHub [2]

Alire Crate: Apdf [3]

Changes in this version:

- Vector graphics: added Arc and Circle.
- Navigation: added Hyperlink methods for producing links within or outside the document.

NB: most of recents additions to the package are contributions from users - kudos!

The main demo (pdf_out_demo.adb) now includes some pie charts and hyperlink examples.

An additional demo shows graphically various steps of the k-means data clustering / unsupervised machine learning method.

[1] https://apdf.sourceforge.io/

[2] https://github.com/zertovitch/ada-pdf-writer

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

From: Shuihuzhuan <Shuihuzhuan@forum.ada-lang.io>
Date: Mon, 24 Feb 2025 13:29:44 +0000

I tried the library 1 or 2 years ago but I was stuck with how to compute the width and height of the text to display. It's not part of the PDF specification, but you need it to layout the text correctly (ellipsize/truncate/split lines/whatever). How would you do that?

From: zertovitch <zertovitch@forum.ada-lang.io>
Date: Mon, 24 Feb 2025 18:35:28 +0000

There is an investigative work behind this topic: get the font metrics.

For the "standard PDF fonts" the PDF 1.7 (ISO standard) says:

> 9.6.2.2 Standard Type 1 Fonts (Standard 14 Fonts)

> The PostScript names of 14 Type 1 fonts, known as the standard 14 fonts, are as follows: Times-Roman, Helvetica, Courier, Symbol, Times-Bold, Helvetica-Bold, Courier-Bold, ZapfDingbats, Times-Italic, Helvetica-Oblique, Courier-Oblique, Times-BoldItalic, Helvetica-BoldOblique, Courier-BoldOblique

> These fonts, or their font metrics and suitable substitution fonts, shall be available to the conforming reader.

> NOTE The character sets and encodings for these fonts are listed in Annex D. The font metrics files for the standard 14 fonts are available from the ASN Web site (see the Bibliography). For more information on font metrics, see Adobe Technical Note #5004, Adobe Font Metrics File Format Specification.

For usable numbers, I'd look into the sources of Ghostscript or PDF.js.

From: Shuihuzhuan <Shuihuzhuan@forum.ada-lang.io>
Date: Mon, 24 Feb 2025 20:41:48 +0000

Thanks for the tip. I'll look at that.

From: zertovitch <zertovitch@forum.ada-lang.io>
Date: Thu, 27 Feb 2025 04:03:49 +0000

There are also open-source PDF generators with that kind of information.

Picking randomly: this [1] or that [2].

[1] https://github.com/DavBfr/dart_pdf/blob/9f54667c6a19e3579495c4a5c185d0642dd6c6fb/pdf/lib/src/pdf/font/type1_fonts.dart

[2] https://github.com/py-pdf/fpdf2/blob/341689600a60296aa5154f5096c12e647e90c511/fpdf/fonts.py

From: zertovitch <zertovitch@forum.ada-lang.io>
Date: Thu, 27 Feb 2025 20:44:08 +0000

There is now a method called Bounding_Box [1] that works for standard PDF fonts - see the main demo for an example.

Some vertical adjustment is still needed…

[1] https://github.com/zertovitch/ada-pdf-writer/commit/4455fc3

From: Shuihuzhuan <Shuihuzhuan@forum.ada-lang.io>
Date: Fri, 28 Feb 2025 07:39:09 +0000

Thank you for this nice addition! It will help a lot.

Alire v2.1

From: mosteo <mosteo@forum.ada-lang.io>
Subject: ANN: Alire v2.1
Date: Tue, 4 Mar 2025 12:08:13 +0000
Newsgroups: forum.ada-lang.io

Release v2.1.0 · alire-project/alire [1]

This is mainly a maintenance release with numerous bugfixes. However, some important internal refactorings and a few new minor features justify releasing it as a new minor version.

Thanks to all contributors, issue reporters, and final users.

[1] https://github.com/alire-project/alire/releases/tag/v2.1.0

Ada Caser v0.1.0

From: simonjwright <simonjwright@forum.ada-lang.io>
Subject: Ada Caser 0.1.0
Date: Sun, 9 Mar 2025 12:25:51 +0000
Newsgroups: forum.ada-lang.io

This is the first public release of ada_caser [1] in Alire. It is a tool to adjust the casing in an Ada source code to a "normal" form. The normal casing form in Ada source has keywords in lower case, identifiers in title case (Title_Case).

If a project file is supplied, ada_caser reads the project file, and the tree of referenced projects. It then uses Libadalang to find the declaration (if any) of each identifier in the input source code and replaces the source text with that of the declaration.

If no project file is supplied, ada_caser uses a default project which allows it to access the standard library (Ada*, GNAT*, Interfaces*).

So, for example, ada.text_io.fixed_io will be replaced by Ada.Text_IO.Fixed_IO.

For previously-declared identifiers, ada_caser will convert the identifier to the casing with which it was declared.

The default processing of undeclared identifiers is to convert the first character of each word to upper-case (reserved words are converted to lower-case, other tokens are left unchanged).

This default can be overridden using casing dictionaries.

Suggestions for improvements welcome!

[1] https://github.com/simonjwright/ada_caser

HAC v0.41

From: zertovitch <zertovitch@forum.ada-lang.io>
Subject: HAC version 0.41
Date: Wed, 12 Mar 2025 19:09:10 +0000
Newsgroups: forum.ada-lang.io

HAC - *H*AC *A*da *C*ompiler - is a small, quick, open-source Ada compiler, covering a subset of the Ada language.

HAC is perhaps the first open-source (albeit partial) Ada compiler fully programmed in Ada itself.

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

Sources, site #1:
https://sourceforge.net/projects/hacadacompiler/

Sources, site #2:
https://github.com/zertovitch/hac

Alire Crate:
https://alire.ada.dev/crates/hac

What's new in this version:

- Improved some compilation diagnostics.
- Improved support for the String type.
- Added the 25 Advent of Code 2024 solutions as examples and regression tests (now 122 in total).
- Added new demos that output PDF files.

GNAT for Target "aarch64-linux" from Host "x86_64-linux"

From: ocw <ocw@forum.ada-lang.io>
Subject: Gnat cross-compilation toolchain for target "aarch64-linux" (from host "x86_64-linux")
Date: Mon, 17 Mar 2025 16:18:10 +0000
Newsgroups: forum.ada-lang.io

Anyone out there has (or can provide) a GNAT FSF toolchain build/release (preferably for the latest 14.2 version) for cross-compilation, for an "aarch64-linux" target, from a "x86_64-linux" host?

I am trying to generate one using the alire-project instructions and scripts [1], but haven't managed to achieve a successful completion yet.

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

From: JeremyGrosser <JeremyGrosser@forum.ada-lang.io>
Date: Mon, 17 Mar 2025 16:23:47 +0000

I don't know about an Alire cross toolchain for that target, but on Debian stable (bookworm) you can apt install gnat-12-aarch64-linux-gnu. GNAT 14 is available in Debian testing (trixie).

From: ocw <ocw@forum.ada-lang.io>
Date: Mon, 17 Mar 2025 16:44:05 +0000

You are right, that will probably be the safest path. Thank you for the quick answer and for taking the blindfold off my eyes (in fact, if the solution had been a dog, it would have bitten me for seeking unnecessary complications)

From: Irvise <Irvise@forum.ada-lang.io>
Date: Mon, 17 Mar 2025 21:46:56 +0000

AFAIK, aarch64 build targets just became available in Github's CI, so hopefully we can see native toolchains being available in Alire by default. But, it may take some time…

From: Irvise <Irvise@forum.ada-lang.io>
Date: Wed, 19 Mar 2025 20:03:19 +0000

Well… That was quick, Aarch64 Linux GNAT [1] It is now merged!

[1] https://github.com/alire-project/alire-index/pull/1440

Zip-Ada v61

From: zertovitch <zertovitch@forum.ada-lang.io>
Subject: Zip-Ada version 61
Date: Tue, 18 Mar 2025 01:13:28 +0000
Newsgroups: forum.ada-lang.io

Zip-Ada is a free, open-source, independent programming library for dealing with the Zip compressed archive file format. It includes LZMA & BZip2 independent compressor & decompressor pairs (can be used outside of the Zip archive context).

Home page: https://unzip-ada.sourceforge.io/

Sources, site #1: SourceForge.net [1]
Sources, site #2: GitHub [2]

Alire Crate: Zipada [3]

Changes in this release:

- BZip2 encoder: improved performance by reducing possible entropy codings chosen by brute-force.
- BZip2 encoder: added segmentation of blocks for heterogeneous data.

[1] https://sourceforge.net/projects/unzip-ada/

[2] https://github.com/zertovitch/zip-ada

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

Ada PDF Writer v8

From: zertovitch <zertovitch@forum.ada-lang.io>
Subject: Ada PDF Writer version 8
Date: Wed, 19 Mar 2025 06:43:51 +0000
Newsgroups: forum.ada-lang.io

[Removed repeated information from version 7. —arm]

Changes in this release:

- Added Bounding_Box method for texts written with standard fonts.

It allows for predicting the room each chunk of text will require on the page.

Further development of that could be to output a text on multiple lines within a rectangle, with hyphenation, flexible spacing, …

GWindows v1.5.0

From: zertovitch <zertovitch@forum.ada-lang.io>
Subject: GWindows version 1.5.0
Date: Mon, 24 Mar 2025 04:58:23 +0000
Newsgroups: forum.ada-lang.io

GWindows is a Microsoft Windows framework for programming GUIs (Graphical User Interfaces) with Ada.

GWindows Alire crate: [1]

GWindows Project site 1: here [2]

GWindows Project site 2: here [3]

- GWindows itself is 100% made with Ada.
- GWindows is free and open-source.
- License: MIT.

Changes in this release:

- Various fixes and additions.
- License change from GMGPL to MIT.

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

[2] https://github.com/zertovitch/gwindows

[3] https://sf.net/projects/gnavi/

Any Experience from Ada/Vulkan?

From: reinert <reinert@forum.ada-lang.io>
Subject: Any experience from Ada/Vulkan?
Date: Tue, 25 Mar 2025 10:29:17 +0000
Newsgroups: forum.ada-lang.io

Out of curiosity, does anyone have experience using Vulkan with Ada? Are there any mature bindings available?

From: jere <jere@forum.ada-lang.io>
Date: Tue, 25 Mar 2025 12:52:00 +0000

- It looks like Lucretia started some here: GitHub - Lucretia/adavulkan: Ada 2012 binding to Vulkan [1]
- Phasercat games has some bindings here: VulkAda - Phaser Cat [2]
- This is more of a side thing, but getintogamedev has been learning Ada to use with Vulkan. I think the first video or two has some vulkan setup stuff: [3]

There may be more things

[1] https://github.com/Lucretia/adavulkan

[2] https://phasercat.com/vulkada/

[3] https://www.youtube.com/watch?v=RoIjGpXVvuw&list=PLn3eTxaOtL2Oxl9HbNOhIPVozAWCeyCZ4&pp=0gcJCWMEOCosWNin

From: Lucretia <Lucretia@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 21:46:11 +0000

Yeah, I started some, but didn't get that far.

From: mgrojo <mgrojo@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 20:19:57 +0000

The AdaDoom3 project has a thin Vulkan binding too.

github.com/AdaDoom3/AdaDoom3

I cannot give an opinion on any of them.

Gprdeps: a New Tool for GPR Project Files Linting

From: ebriot <ebriot@forum.ada-lang.io>
Subject: Gprdeps: a new tool for gpr project files linting
Date: Mon, 31 Mar 2025 15:54:44 +0000
Newsgroups: forum.ada-lang.io

I have developed (the beginning of) a new tool, written in Rust. It is meant to replace a tool that my company DeepBlueCapital has developed internally and which was initially written in Python. But the Rust version is 10 or 15 times faster.

Dependency-graph build for AdaCore's GPR files [1]

It starts by loading a GPR file and its dependencies (or all project files in a given directory and child directories). It then finds all source files for all the projects. So far this is similar to what gprbuild itself does, but gprdeps does this loading for all scenarios at the same time (so it knows sources for debug and release mode, or for tasking and non-tasking, and so on depending on which scenario variables you are using).

It then builds a large internal graph to represent project dependencies and file dependencies (which it gets by parsing Ada and C source files, not using ALI files like gprbuild does). So as soon as we finish loading, we have this big graph for all possible combinations of scenarios. gprbuild never builds such a graph, and only loads for one scenario. Also, did I mention speed? On our 2 million lines of sources, the tool loads everything, including parsing all source files, in 0.3s on my laptop. gprbuild takes about 10s before it spawns any gnat (which is basically the only thing we can measure)

Once we have this graph, we can perform a number of queries:

- given a source path, report all source files it (possibly indirectly) depends on
- given a source path; report all source files that depend on it
- given two paths, report the shortest dependency path between the two. Useful to understand why a file ends up in the closure and has to be elaborated, for instance
- report unused source files; we have discovered quite a number of old files that were not used anymore for any executable or any compilation mode, but were still compiled by gprbuild when using -u for instance
- find exactly which compilation switches would be used for a source file in any scenario

This is very much work in progress, and could be used as the basis for a better and faster gprbuild for instance, or maybe in the context of Alire. Unfortunately, I will not be able to maintain it in the long term, so this tool is looking for maintainers if we are to add new features.

It is already useful as is though, so I encourage you to have a go.

[1] https://github.com/briot/gprdeps

Ada and Other Languages

Trouble Interfacing with C libraries

From: Trescott <Trescott@forum.ada-lang.io>
Subject: Trouble interfacing with C libraries; Making thick bindings
Date: Fri, 14 Feb 2025 09:31:53 +0000
Newsgroups: forum.ada-lang.io

Background

While I've used Ada for work projects, those projects did not interface with C code and did not use access types. So, I'm lacking experience needed for my personal projects that interface with C libraries.

When trying to learn how to interface with C code, the various tutorials seemed like they would be helpful. But once I faced real-world code, I realized the tutorials did not cover examples of what I needed to do.

During the process of creating this post, I solved the problems, that prevented compiling, that I originally intended to include as questions. But this still stands as an example of some of the difficulties new Ada users face. The Ada ecosystem is not as comprehensive as, for example, Python with its "batteries included" philosophy. So, new Ada users are likely to need to interface with C libraries.

This C library function, that I needed to interface with, used the common C practice of taking a "buffer" argument to output to. I could not find any example of interfacing with this kind of function. So I was left to decipher the Ada Reference Manual's Appendix B.3.1 "The Package Interfaces.C.Strings" [1]. Which didn't really provide any help with how to /*use*/ it.

Although my original intent was to get help making this code compile, I am still posting it because it may help others who are facing a similar problem. I also do have other (less important) questions about the code I wrote.

Problem

For a Linux utility, I need to read the target of symbolic links (symlinks). GNAT.OS_Lib has a function Is_Symbolic_Link that I can use to verify a file is (or is not) a symbolic link, but it doesn't have a function to read the target of the link (what it links to). I've tried to find an existing Ada package that has this function, but I have not found one.

So, I need to create a binding to the POSIX function. The function readlink in unistd.h has the following signature:

ssize_t readlink(const char* restrict path, char* restrict buf, size_t bufsize);

Seems like it would be straightforward to use…

Auto Generating the Thin Binding

I used g++ to generate thin bindings to the unistd.h library:

g++ -c -fdump-ada-spec -C /usr/include/unistd.h

This produced bindings to everything in that file and all its dependencies, creating the following files (in no particular order):

- unistd_h.ads
- stddef_h.ads
- bits_unistd_ext_h.ads
- bits_types_h.ads
- bits_getopt_core_h.ads
- bits_confname_h.ads

Most of these I won't need, but it doesn't hurt to leave them.

The thin binding generated for readlink is as follows:

```
   -- Read the contents of the symbolic link
   -- PATH into no more than
   --  LEN bytes of BUF.  The contents are not
   -- null-terminated.
   --  Returns the number of characters read,
   -- or -1 for errors.
   function readlink
     (uu_path : Interfaces.C.Strings.chars_ptr;
      uu_buf  : Interfaces.C.Strings.chars_ptr;
      uu_len  : stddef_h.size_t) return ssize_t
      -- /usr/include/unistd.h:838
   with Import        => True,
        Convention    => C,
        External_Name => "readlink";
```

Creating a Thick Binding

The trouble of creating a thick binding is the same as the trouble of using the thin binding. So, even if I was to use this thin binding directly, I would be having the same problem(s).

Concept of use:

- A string containing the path/name of the symlink file is passed in.
- A pre-allocated string buffer is used to pass out the target path/name.
- This is where I was struggling.
- An integer is used to pass in the size of the buffer.
- An integer is returned as the size of the string put in the buffer.

An Attempt at a Thick Binding:

```
function Read_Link(filename: String) return
String is
   path : Interfaces.C.Strings.chars_ptr :=
   Interfaces.C.Strings.New_String(filename);
   -- path/filename of the symlink to read
   buf : Interfaces.C.Strings.chars_ptr :=
   Interfaces.C.Strings.New_String("How do I
   allocate a large size without a very long
    literal?");
   -- Pre-allocated string (buffer) to receive the
   -- target path/filename
   Len : stddef_h.size_t := buf'size;
   -- Size (length) of the buffer; Maximum
   -- length that can be used for the target
   --path/filename

   Used : unistd_h.ssize_t;
   -- How many characters of "buf" were used,
  --  or an error code if negitive
begin
   -- Make the call to the system library:
   Used := unistd_h.readlink(path, buf, Len);
   declare
      Target : String(1..Integer(Used));
      -- Return variable
   begin
      -- If "used" is within the range of "buf":
      if Used > 0 and then Used
```

This compiles and seems to run successfully.

Questions:

- I think New_String is allocating memory and making a copy of the string. But in this use case, the string, filename, on the stack will live for the life of the call to readlink so I would like to pass that existing string. Can I do that cleanly?

- buf is being created using a literal string. I would prefer to just specify the bounds or size. Can I do that? And how?
- /Bonus question/: Is there anything else wrong with this code?

Thank you for taking the time to read this post and consider my questions.

[1] https://docs.adacore.com/live/wave/arm12/html/arm12/arm12-B-3-1.html

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Fri, 14 Feb 2025 10:33:41 +0000

Well, chars_ptr is used when you deal with dynamically allocated C "strings" or with ones returned from C. When arrays are managed by you, you should use char_array.

[...]

From: Lucretia <Lucretia@forum.ada-lang.io>
Date: Fri, 14 Feb 2025 10:50:27 +0000

Binding generators /can/ help IF they have the meta information for each parameter, but they don't.

i.e. this pointer parameter, is it:

- const?
- an in or an out?
- an array?
- returning a pointer to something (** and ***)?
- is it filling an array?
- etc.

From: Trescott <Trescott@forum.ada-lang.io>
Date: Mon, 17 Feb 2025 18:49:51 +0000

Thank you everyone for your responses! This has been very helpful.

What I'm getting from you all is that (maybe over simplified here):

- The auto generated bindings are not the best.
- chars_ptr is for dynamically allocated C-Strings originating from C code.
- char_array is for C-Strings originating from Ada code.

> do not tell us fairy tales about Python!

I couldn't help but to make the comparison because this project is taking a Python script I wrote a while back, and converting it to Ada. The python library had this function and Ada didn't.

I originally threw this script together for fast results. It worked for the expected case, but wasn't designed to handle anything outside the expected case. A few years later, I try to use it again and the environment changed somewhere, shifting the norm and breaking my script.

Looking at my script, years later, I find the code to be an unintelligible mess, as would be expected for the lack of effort. So, I decided it's time to do it right. Hence Ada.

I did fix the original Python script, but it's time to do it right.

> I think you can instead use the switch -fdump-ada-spec-slim to only generate bindings for the specified file (and not all the others).

Thanks @jere. I wasn't too concerned about all the files that were generated, I was just trying to be complete in my documentation for future readers. But this will be handy for the next time I use the auto generate bindings feature.

C/C++ VS. Ada Rant

One of the aspects of Ada that I love is how it can create complete abstractions; where the specification is sufficient to correctly use the interface without needing to rely on the documentation or peeking behind the curtain at the implementation.

In contrast, C and its related languages are, as @dmitry-kazakov said, ill-defined. The "specification" (header) leaves out necessary information that must then be supplemented with documentation (but often is not).

Some of the things left to be identified and explained in the documentation include:

- is a pointer for
- a null-terminated string
- a non-terminated string
- an array
- something else
- how is the length/size specified for an array or non-terminated string
- Is a parameter an "/in/", "/out/", or "/in out/" parameter

It seems that inevitably something is missing in the documentation and you must peek behind the curtain at the implementation to see how it works in order to know how to interface with it.

Unfortunately, the badness of C bleeds into Ada when interfacing with C. The Interfaces.C package gives the tools for the job, but how to use them relies on some knowledge not capture in the interface itself and the existing documentation is not sufficient.

Improving documentation

At some point in the near-ish future, I would like to add this info to the documentation at (Tutorial | ada-lang.io, an Ada community site [1]), but I don't understand this topic well enough to do it right by myself.

I could rough out a draft, if others could correct the content and suggest missing parts. I could then handle polishing up the final draft.

If we are going to grow the Ada ecosystem, we need more developers who are familiar with and comfortable working with C interfacing tasks. So, I think it is important to give this topic comprehensive documentation.

Would anyone be willing to work with me on this?

[1] https://ada-lang.io/docs/category/tutorial

From: Lucretia <Lucretia@forum.ada-lang.io>
Date: Mon, 17 Feb 2025 19:52:15 +0000

You can look at SDLAda [1] and its associated libs as they are mostly written by hand, they should give you an idea of how it should be done.

[1] https://github.com/ada-game-framework/sdlada

Ada Practice

Array of Discriminated Tasks

From: evanescente-ondine <evanescente-ondine@forum.ada-lang.io>
Subject: Array of discriminated tasks
Date: Tue, 14 Jan 2025 13:15:53 +0000
Newsgroups: forum.ada-lang.io

Since there are limited aggregates now, I wonder why we can't have task aggregates and protected aggregates as well? They would be treated just the same. Records with only discriminants as components.

They can't be returned as local objects, nor be passed a local access value, and if put in an extended return block, they would activate there for some processing before returning to the calling block (whereas as a variable on the stack as through an allocator). Are there other issues?

edit: Equality isn't defined on tasks. But it doesn't make sense on records containing tasks either.

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Tue, 14 Jan 2025 13:58:37 +0000

OK, since I had this issue many, many times, I just give you an advice you can ignore.

You should use access to task, always. There are many factors at play and discriminants is only one of many:

- Parameters. Add an entry Start and pass parameters when you create a task.

- Discriminants. You usually need an access to the container object which precludes arrays etc.
- Task termination. The terminate alternative is almost useless. You can use it in a few very rare cases, but normally you cannot. Terminate does not mix with timed selective accept and you cannot use it [with] entry calls. So you usually need a Stop entry accepted at the end of the task body loop.
- Controlled types encapsulating tasks do not fly because Initialize is called when tasks are running and Finalize is called when tasks have been stopped. That makes an access to task a must. You want to start them from Initialize and stop from Finalize. The language choice is safe and consistent with the model finalization of components, but unusable for all practical purposes [*].
- Stopping and deallocating a task. The template is:

```
Worker.Stop;
-- Ask the worker to terminate
while not Worker'Terminated loop
-- Check if it did
   delay 0.001;
end loop;
Free (Worker); -- Now I can free it
```

I do not know if the latest Ada standard fixed the issue, but in older Ada you would have a leak if you deallocate a running task.

[*] The problem with task components and task access discriminants to a class-wide container is the same as the problem of class-wide constructors in general. I do not go into details, but Initialize/Finalize hack is a class-wide constructor and component task start/stop is a type-specific constructor. You cannot reorder them, so the language choice. Until the language provides both constructors user-defined, the problem will stay.

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Tue, 14 Jan 2025 14:41:20 +0000

> Stopping and deallocating a task. The template is:

This was my solution, too.

I have a feeling that the issue was fixed in GNAT a few years ago, but don't remember when.

From: ebriot <ebriot@forum.ada-lang.io>
Date: Wed, 15 Jan 2025 13:47:04 +0000

I am using the same solution for freeing access to tasks as Dmitry and Simon, I did a blog post a while ago because as far as I know the issue still exists in GNAT (Freeing task-access objects – Deep Blue Capital [1]) The blog describes the same solution, but shows why it is useful if people want to understand the context a bit more.

[1] https://deepbluecap.com/freeing-task-access-objects/

From: jere <jere@forum.ada-lang.io>
Date: Wed, 15 Jan 2025 19:09:03 +0000

I think the use of a default was just to give a means to tell the compiler to pick a universal size that works for all (for some compilers like Janus, this means they pick the size of a pointer and use the heap to allocate the structure. GNAT chooses to just use the size of the largest variant, similar to how a union would be implemented). In absence of that, they wanted the compiler to pick the appropriate size for the type.

Consider the following:

```
type Integer_Array is array(Positive range )
of Integer;
type Vector(Capacity : Natural) is record
  Elements : Integer_Array(1..Capacity);
end record;
```

or

```
type Thing(Double_Size : Boolean) is record
  case Double_Size is
    when False => Small :
                  Integer_Array(1..10);
    when True  => Large :
                  Integer_Array(1..20);
  end case;
end record;
```

In those cases the size can be multiple different things depending on the discriminant provided. So the compiler can't tell the size of that type at compile time. Yes in a simple case it should be easy, but the language allows for much more complex cases so the rules were set up to allow for all cases. There's definitely an argument to be made that maybe the language rules could provide more flexibility though if the type is simple. It's definitely something you could bring up at the ARG's github repo via an issue to see if they would make a change to the language.

Side note: GNAT doesn't support them fully but there are also coextensions:

```
type Thing(Coextension : not null access
Complex_Type) is record
  -- other stuff
end record;
v : Thing(new Complex_Type'(initialization));
```

Here the language requires the Coextension item being created here to be created on stack along with v instead of on the heap (as the new might make you think). This is another scenario where the size of the type may not be known at compile time.

From: jere <jere@forum.ada-lang.io>
Date: Thu, 16 Jan 2025 00:13:31 +0000

I was thinking that in order to have the same lifetime, they would need to be allocated in the same way, so if one is allocated on the stack, the other is too, if one is allocated on the heap, the other is too, etc. In my example I was saying v was allocated on the stack, so the coextension would need to be allocated the same way (on the stack) in order to preserve the same lifetime rule. I could be wrong, but that was the thought process. I wasn't saying all coextensions were on the stack.

Also note this part from the AARM:

3.10.2

> *Ramification:* The rules of access discriminants are such that when the space for an object with a coextension is reclaimed, the space for the coextensions can be reclaimed. Hence, there is implementation advice (see 13.11) that an object and its coextensions all be allocated from the same storage pool (or stack frame, in the case of a declared object).

EF: Operations of Access Types [1]

and 13.11:

> If the allocator [2] is defining a coextension (see 3.10.2 [3]) of an object being created by an outer allocator [2], then the storage pool used for the outer allocator [2] should also be used for the coextension;

REF: Storage Management [4]

[1] http://www.ada-auth.org/standards/22aarm/html/AA-3-10-2.html#I2646

[2] http://www.ada-auth.org/standards/22aarm/html/AA-4-8.html#S0164

[3] http://www.ada-auth.org/standards/22aarm/html/AA-3-10-2.html

[4] http://www.ada-auth.org/standards/22aarm/html/AA-13-11.html

Code of Protected Actions Thread of Control

From: evanescente-ondine <evanescente-ondine@forum.ada-lang.io>
Subject: Code of protected actions can be executed by any thread of control?
Date: Thu, 16 Jan 2025 15:30:29 +0000
Newsgroups: forum.ada-lang.io

Can you explain the following to me?

> An implementation may perform the sequence of steps of a protected action using any thread of control; it need not be that of the task that started the protected action (RM)

> (book) Assume that the producer in the program using protected objects attempts to append an item to a full buffer. The entry call Buffer.Append will be enqueued, the producer task will be blocked and a consumer task will be allowed to execute. Eventually, the consumer task will take an item from the buffer. Upon completion of the body of Take, servicing of the entry queue can be done by the consumer

task, as shown in the following diagram:

If the caller of the protected action does not execute the code, who does? It can't be arbitrary, and I can't see why a completely unrelated thread would do it. I mean, when they execute on the same core it would be irrelevant, since in any case the calling task is blocked until completion of the action, no matter where it happens. But it matters in the case of multicore, multiprocessor or fully distributed systems, right?

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Thu, 16 Jan 2025 15:55:51 +0000

> If the caller of the protected action does not execute the code, who does?

The other task (the consumer), as citation explained. You do not need to switch back to the caller (the producer) in order to execute the action.

> I mean, when they execute on the same core it would be irrelevant

It is very relevant because you do not need to switch the context, which is expensive. The consumer task already owns the processor. It executes the action and continues with the data. The producer has no processor, but unblocked and can continue when it gets the processor back.

The idea is to execute the protected action by any involved task if the task is active. This reduces latency to zero.

> But it matters in case of multicore

Even if the tasks run on different cores they share memory. So some advantage is still there.

Then there is a requeue statement…

Using an Ada 2022 Spec from an Ada 2012 Client

From: mosteo <mosteo@forum.ada-lang.io>
Subject: Using an Ada 2022 spec from an Ada 2012 client
Date: Sat, 18 Jan 2025 13:03:25 +0000
Newsgroups: forum.ada-lang.io

(TL;DR: I came at this with the impression that aspects, just like pragmas, could be ignored when unknown. This seems not to be mandated, but only allowed, by the ARM 2022.)

The idea I was attempting is that a library can use Ada 2022 features but an Ada 2012 client can still "with" it, without maintaining two sets of sources, by ignoring unknown pragmas.

However, "future" aspects are reported as an error (using GNAT 14.1):

```
pragma Ada_2012;
package Multiver is

   type Modern is null record
     with Aggregate => – line 6
          (Empty      => Empty,
           Add_Unnamed => Drop),
        Integer_Literal => From_Int, – line 8
        Unknown_Aspect; – line 9

   function Empty return Modern is
          (null record);
   procedure Drop (X : in out Modern;
             I : Integer) is null;

   function From_Int (X : String) return
            Modern is (Empty);

end Multiver;
```

Output:

multiver.ads:6:24: error: incompatible with Ada version set at line 1

multiver.ads:9:11: warning: "Unknown_Aspect" is not a valid aspect identifier [enabled by default]

That's a self-contained example, but the same happens in the intended usage with two separate projects, library using -gnat2022 and client withing Multiver using -gnat12 and no Pragma Ada_2012 in the source.

Note:

- Expected warning on line 9
- Error instead of warning on line 6
- Unexpected silence about line 8, which is maybe a bug.

So the thing is: if that error could be downgraded to a warning, I could have my cake and eat it.

(Just tested that the same happens with all GNATs in Alire, from 11 upwards. GNAT 10 rejects all unknown aspects).

After some digging, I find that the ARM 2012 says nothing about unknown aspects. The ARM 2022 gives permission to ignore unknown aspects: 13.1 Operational and Representation Aspects | Ada Programming Language [1], but does not mandate it so…

This seems to come from Version 1.4 of ai12s/ai12-0389-1.txt [2]

So, after all that, I think it boils down to: is there some GNAT switch to enable ignoring all unknown aspects? Not practical for me anymore, since older GNATs would likely not have it, but at this point I'm just curious. And the lesson learned is that aspects are not just syntactic sugar on top of pragmas.

[1] https://ada-lang.io/docs/arm/AA-13/AA-13.1#p39_5_13.1.1

[2] http://www.ada-auth.org/cgi-bin/cvsweb.cgi/ai12s/ai12-03891.txt?rev=1.4&raw=N

From: OneWingedShark <OneWingedShark@forum.ada-lang.io>
Date: Sat, 18 Jan 2025 15:59:34 +0000

> I find that the ARM 2012 says nothing about unknown aspects.

I was sure that there was permission to ignore unknown aspects in Ada2012.

Maybe I'm misremembering. (I *know* there's permission for ignoring unknown pragmas.)

> And the lesson learned is that aspects are not just syntactic sugar on top of pragmas.

There are some which /are/, though. (By definition.)

And that's where the confusion can come from.

From: mgrojo <mgrojo@forum.ada-lang.io>
Date: Sat, 18 *Jan 2025 16:14:04 +0000*

> The ARM 2022 gives permission to ignore unknown aspects [...] but does not mandate it so…

Then, adding any of the SPARK implementation-defined aspects to an Ada program could render it unportable to other compilers, if they choose to reject unknown aspects? I didn't expect that, I thought aspects were like pragmas in that regard.

From: JC001 <JC001@forum.ada-lang.io>
Date: Sun, 19 Jan 2025 10:13:37 +0000

ObjectAda used to treat SPARK aspects as errors. When I pointed out that it is not impossible that someone might want to use OA to compile code proven with SPARK, they changed it to ignore SPARK aspects, so I presume that means it's OK for an Ada-12 compiler to ignore aspects it doesn't implement. I think it still treats non-SPARK unknown aspects as errors.

From: mgrojo <mgrojo@forum.ada-lang.io>
Date: Sun, 19 Jan 2025 12:31:16 +0000

In my opinion, all Ada compilers should implement a switch to ignore all the unknown aspects (and maybe only raise a warning). Otherwise, the permission to add implementation-defined aspects breaks the portability of the standard language. That didn't happen with pragmas.

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Sun, 19 Jan 2025 12:46:04 +0000

IMO, Ada should apply strict rules regarding implementation of the standard. Long ago you could not call a compiler Ada compiler if you did not implement all da standard.

As for pragmas, that is not so simple. Some pragmas are more pragmas than others, e.g. pragma Atomic, pragma Import, representation pragmas etc.

The problem is that both pragmas and aspects (the difference is in the syntax only) can influence the semantics or be just a compiler hint/correctness annotation. You cannot ignore them in the former case. If only the language clearly separated the cases…

From: mgrojo <mgrojo@forum.ada-lang.io>
Date: Sun, 19 Jan 2025 12:43:21 +0000

For GNAT, I've found this:

NF-T622-002 Unknown aspects ignored (2020-06-22)

By default, GNAT will now ignore unknown aspects and generate a warning instead of rejecting them with an error, to provide a handling similar to unknown pragmas and better compatibility across different Ada and GNAT compiler versions.

From https://docs.adacore.com/R/relnotes/features-21

So no switch, in principle, is necessary, but it seems that known aspects belonging to a future Ada version have a different logic, and they aren't considered "unknown".

'Reduce at Compile Time

From: jklmnn <jklmnn@forum.ada-lang.io>
Subject: Sum of an entire array in specification package in compilation time
Date: Wed, 5 Feb 2025 13:24:21 +0000
Newsgroups: forum.ada-lang.io

The problem here is that the result of 'Reduce is not static, even if all its inputs are. So any expression involving a result of 'Reduce cannot be static.

E.g.

```
Another_Table :
Table (1 .. Values'Reduce ("+", 0));
```

will never be static and therefore can only be evaluated at runtime.

For this to work there needs to be a compiler optimization on 'Reduce that understands its purpose and is able to execute it. While it may sound easy for a sum, the other part is that the first argument of 'Reduce is a reference to a function. I'm not too familiar with 'Reduce itself but from what I can tell the function can be anything:

```
Non_Const : Integer := 42;
function Non_Const_F (L, R: Integer) return Integer is
  (L + R + Non_Const);
type I_Array is array (range ) of Integer;
Const_Array : constant I_Array (1 .. 10) := (1, ..., 10);
Const_Var : constant Integer := Const_Array'Reduce (Non_Const_F, 0);
```

While all direct inputs to 'Reduce are theoretically static the reducer has non-static inputs. The same could be true for side effects for example.

I'm not saying that a compiler optimization that can do it is impossible, but it certainly isn't trivial, which is probably one of the reasons this doesn't work.

From: jere <jere@forum.ada-lang.io>
Date: Wed, 5 Feb 2025 14:39:56 +0000

EDIT: Just to clarify, this is not a disagreement with your post, just wanted to toss out some clarifying stuff below. You touched on it obviously as well.

> [example] will never be static and therefore can only be evaluated at runtime.

Just wanted to clarify that for new readers this statement isn't true in general (but can be true in specific cases). It is true that since Reduce is not static, that it is never guaranteed to be static. That said, the compiler can (and currently does sometimes) choose to optimize it statically. Consider the test code:

```
procedure Example is
  type Table is array(Positive range <>) of Integer;
  Values : constant Table(1..10) := (others => 1);
  -- Final index should be 10 after the
  -- Reduce calculates
  Another_Table : Table (1 .. Values'Reduce ("+", 0)) := (1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
  Value : Integer := Another_Table(10) with Volatile;
begin
  null;
end Example;
```

with compiler switches -O3 -gnat2022 the compiler is able to figure out the last index of that table is indeed 10 at compile time (not run time) and emits:

```
_ada_example:
mov     DWORD PTR [rsp-12], 10    # value,
ret
```

Without needing to verify that the bounds of the array match the size of the initialization aggregate expression.

And if you change it to

```
procedure Example is
  type Table is array(Positive range <>) of Integer;
  Values : constant Table(1..10) := (others => 0);
  -- Final index should be 10 after the
  -- Reduce calculates
  Another_Table : Table (1 .. Values'Reduce ("+", 0)) := (1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
  Value : Integer := Another_Table(10) with Volatile;
begin
  null;
end Example;
```

The compiler replaces it with:

```
.LC0:
        .ascii  "example.adb"
        .zero   1
_ada_example:
        sub     rsp, 8    #,
        mov     esi, 9    #,
        mov     edi, OFFSET FLAT:.LC0     #,
        call
        __gnat_rcheck_CE_Length_Check  #
```

which throws a constraint error since the array is actually a length of 0 with the change

From: jklmnn <jklmnn@forum.ada-lang.io>
Date: Wed, 5 Feb 2025 14:56:15 +0000

You're right, but™ there is a pitfall here, unfortunately. I tried your example to check whether it can be used to solve @daniel's original question. The problem we're facing here is that there are two optimization steps. The behavior you see is coming from the second optimization step but for the task allocator to work statically it needs to be in the first optimization step. If I expand the array declaration from your example I get:

```
  R13b : constant integer :=
    do
      B7b : integer := 0;
      L10b : for C11b in 1 .. 10 loop
        E8b : integer renames values (C11b);
        B7b := B7b {+} E8b;
      end loop L10b;
    in B7b end;
  subtype example__Tanother_tableS is example__table (1 .. R13b);
```

compiled with

```
gcc -c -x ada -gnatA -gnat2022 -O3 -g example.adb
```

What (I think) happens here is that GCC optimizes the local loop generated by the frontend and hence you can see the static value in the generated binary. The problem is the restriction No_Dynamic_Task_Allocation is checked *before* that happens. This also isn't easily solvable because the frontend would need to figure out whether GCC can optimize it out to or not.

So your example works, but only for a different kind of static

Note: I'm not sure how much optimization is actually done in the expansion step, but if there is any it's not sufficient for 'Reduce, at least not at the time.

Use of @ and Implicit_Dereference

From: Lucretia <Lucretia@forum.ada-lang.io>
Subject: Is this a compiler bug? Use of @ and Implicit_Dereference
Date: Wed, 19 Feb 2025 12:55:02 +0000
Newsgroups: forum.ada-lang.io

I have the following code:

```
Model_View_Stack.Top := @ *
Matrix4s.Translate (0.0, 0.0, 0.0);
```

Where Top [1] returns a user defined reference with Implicit_Dereference [2], but compilation gives the following error:

```
simple_solar_system.adb:171:07: error:
actual for aliased formal "Self" must be
aliased object
simple_solar_system.adb:171:23: error:
ambiguous left-hand side in assignment
simple_solar_system.adb:171:33: error:
invalid operand types for operator "*"
simple_solar_system.adb:171:33: error: left
operand has private type "Reference_Type"
defined at maths-stacks.ads:16, instance at
maths-matrix4s-stacks.ads:8
simple_solar_system.adb:171:33: error: right
operand has type "Matrix4" defined at maths-
matrix4s.ads:58
```

[Similar errors omitted. —arm]

Shouldn't the compiler be able to work out that @ is being dereferenced?

Edit: So, even wierder.

The following compiles (once the Model_View_Stack is made aliased):

```
Maths.Matrix4s.Stacks.Top
(Model_View_Stack) :=
Maths.Matrix4s.Stacks.Top
(Model_View_Stack) * Matrix4s.Translate
(0.0, 0.0, 0.0);
```

But not with @.

This works:

```
Model_View_Stack.Top :=
Model_View_Stack.Top * Matrix4s.Translate
(0.0, 0.0, 0.0);
```

[1] https://github.com/ada-game-framework/maths/blob/11fcfec4944074d92e29899b2513941a50e7a3a5/src/utils/maths-stacks.ads#L31

[2] https://github.com/ada-game-framework/maths/blob/11fcfec4944074d92e29899b2513941a50e7a3a5/src/utils/maths-stacks.ads#L16

From: jere <jere@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 13:53:07 +0000

It seems like a compiler bug to me (though maybe there is some arcane rule). I see you tested casting the value. You might also try qualifying it or also a renames object and see if those also fail/work.

From: Nordic_Dogsledding <Nordic_Dogsledding@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 16:58:34 +0000

Without a reproducer, there is not much to say.

However, @ is not a kind of C makro, a literal replacement. It seems that the left and right sides of Model_View_Stack.Top have different resolutions - one perhaps an access value and the other an implicit dereference.

From: sttaft <sttaft@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 17:14:18 +0000

The interaction between @ and implicit dereference is currently being reviewed by the Ada Rapporteur Group (ARG). There do seem to be some complexities here! I anticipate that we will create an ARG GitHub issue shortly on this topic (issues can be found at [1]).

In any case, it is true that @ is not a macro, but rather requires the LHS of the assignment to be evaluated to identify an object, and the @ represents a reference to that same object. But exactly when and where in these contexts implicit dereference gets applied needs to be further clarified in the Ada RM.

[1] https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues

From: Lucretia <Lucretia@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 17:47:12 +0000

Why are people talking about C macros? I never mentioned that.

The source is linked and wouldn't be difficult to make a small reproducer.

From: Nordic_Dogsledding <Nordic_Dogsledding@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 18:21:23 +0000

> Why are people talking about C macros?

Well, people write about casts where there are no casts in Ada, they write about classes which are not a class in Ada. So you shouldn't feel offended when foreign words are used.

Here, this alien wording was used to explain that @ is not a literal replacement of the LHS. In

```
X := @ * ABC;
```

@ evaluates identical to the LHS, whereas in

```
X := X * ABC;
```

X at the RHS might evaluate to something different depending on context.

From: OneWingedShark <OneWingedShark@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 20:04:24 +0000

Another way to say this: The @ is not textual replacement, but a reference to the object on the left-hand side of the assignment. The question/possible-bug here is that this reference's dereference should contain the Implicit_Derefrence attribute of the referenced object and/or implementation thereof and interactions of said dereference.

From: Lucretia <Lucretia@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 20:22:14 +0000

I never expected it to be textual replacement either. I expected it to take the lhs of the assignment, and link via the ast to that on the rhs of the assignment.

From: OneWingedShark <OneWingedShark@forum.ada-lang.io>
Date: Wed, 19 Feb 2025 20:40:18 +0000

Right; if the LRM were to define it on the AST/IR level it would be something like:

/"The @-node shall be a descendent of an ASSIGNMENT-node, and shall represent the object of the ASSIGNMENT-node to which the value will be assigned."/

Pack Aspect with Aliased Component

From: Blady <p.p11@orange.fr>
Subject: Pack aspect with aliased component.
Date: Sun, 2 Mar 2025 09:29:28 +0000
Newsgroups: comp.lang.ada

With the following AARM example:

```
14.e/3{AI05-0229-1}
procedure Q is
   use P1, P2;
   type Array1 is array(Integer range <>)
   of aliased S1
   with Pack; -- warning: cannot pack aliased
components (RM 13.2(7))
```

GNAT issues the warning in the comment above.

Mentions of aliased in following AARM paragraphs are not so clear for me:

13.2 Packed Types

7.1/4 * {AI12-0001-1} Any component of a packed type that is of a by-reference type, that is specified as independently addressable, or that contains an aliased part, shall be aligned according to the alignment of its subtype.

9.c/3 {AI95-00291-02} {AI05-0229-1} Added clarification that the Pack aspect can ignore alignment requirements on types that don't have by-reference or aliased parts. This was always intended, but there was no wording to that effect.

What is correct?

Why packed type couldn't have aliased components?

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 2 Mar 2025 11:33:30 +0000

This is a warning, not an error. The compiler has accepted the declaration and you can use it.

Remember that Pack is a suggestion to the compiler, not a requirement, and the compiler can ignore it, or pack things less tightly than you might like.

An aliased component may be accessed through an alias (access value), and that access will not know that the component has been packed, and so access memory belonging to other components as well. For this reason, aliased components cannot be packed.

You can specify Component_Size for the type; the compiler must either be able to make the components that size, or reject the declaration. Specifying a Component_Size for an aliased component that makes the component smaller than a stand-alone object of the type will be rejected.

From: Blady <p.p11@orange.fr>
Date: Thu, 6 Mar 2025 20:52:44 +0000

Thanks Jeff, your explanation seems clearer to me than the AARM ;)

Fixed Point Numbers

From: MarioBlunk <MarioBlunk@forum.ada-lang.io>
Subject: Fixed Point Numbers
Date: Fri, 7 Mar 2025 18:01:45 +0000
Newsgroups: forum.ada-lang.io

If I've understood everything right, then a statement like

```
for type_distance'small use 0.01;
```

seems essential in order to avoid errors adding up even with binary fixed point numbers.

I've put together a working example to reproduce the problem [1]

```
-- This is a simple ada program,
-- that demonstrates an issue with ordinary
-- fixed point types.
-- This type is also called "binary fixed point
-- type".
with ada.text_io;    use ada.text_io;
procedure demo is
   -- This requested step width is not a power
  -- of two:
   step_width : constant := 0.1;
   -- The nearest step width that is both:
   -- a power of two AND
   -- is less than or equal 0.1
   -- is 0.0625.
   -- So a delta of actually 0.0625 applies for
   -- our fixed point type.
  type type_distance is delta step_width
  range 0.0 .. 100_000.0;
   -- for type_distance'small use step_width;
      distance : type_distance := 0.0;
```

This file has been truncated. See original [1]

So my question is: What value has a binary fixed point type without the statement to use 'small (see above)?

[1] https://github.com/Blunk-electronic/ada_training/blob/c8657f1c2ec6541dc9991d925b65357715e85383/src/fixed_point_numbers/binary_3/demo.adb

From: MarioBlunk <MarioBlunk@forum.ada-lang.io>
Date: Sat, 8 Mar 2025 18:06:31 +0000

Software engineers seldom have insights into what is going on in the hardware industry. A lot of machinery for milling, sawing, drilling, and plotting requires fixed point numbers. Errors adding up are quite dangerous there. So there are good reasons for fixed point numbers.

I've studied a lot of books on Ada programming but it seems I'm the first who is asking the question (see initial post).

From: JC001 <JC001@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 11:27:45 +0000

> it seems I'm the first who is asking the question

Hardly. I saw a program like this in 1984 when I was learning Ada

```
with Text_IO;
procedure Fixed_Problem is
  type Money is delta 0.01;
  Value : Money := 0.0;
begin
  for I in 1 .. 5 loop
    Value := Value + 0.01;
  end loop;
  Text_IO.Put_Line (Item =>
  Money'Image (Value) );
end Fixed_Problem;
```

which output 0.04. The instructor presented it to demonstrate this issue, along with the fix of adding

```
for Money'Small use 0.01;
```

Perhaps Ada teaching materials don't usually mention this. Barne's /Programming in Ada 2012/ says "The implemented values of a fixed-point type are multiples of a positive real number /small/. The actual value of /small/ chosen by the implementation can be any power of 2 less than or equal to D [the delta].", while Clark's /Programming in Ada®: A First Course/ (1985) doesn't mention fixed-point types at all. Booch's /Software Engineering with Ada®/ (second edition, 1987) doesn't mention that the small defaults to a power of 2.

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Fri, 7 Mar 2025 22:41:07 +0000

There wouldn't be errors if the delta was a power of two (e.g. for an LPS25H barometric pressure sensor the LSB in the pressure register is 1/4096 hPa).

I'm not sure why one would use fixed point unless there was some external consideration like that.

Even where the delta is a power of two, there's nothing wrong with stating the small for confirmation.

From: sttaft <sttaft@forum.ada-lang.io>
Date: Sat, 8 Mar 2025 00:36:28 +0000

It is a good question why the Small of a fixed-point type doesn't default to the specified Delta. That was a choice made back in the early days of Ada (e.g. 1980) but in retrospect it seems like an odd choice. I would recommend you pretty much always specify the Small. These days the "aspect specification" syntax makes it pretty easy:

```
type D is delta XXX range AAA .. BBB
with Small => XXX;
```

Yes, it is a bit of the department of redundancy department…

From: Irvise <Irvise@forum.ada-lang.io>
Date: Sat, 8 Mar 2025 22:42:09 +0000

Also, we have decimal point types, which may be better depending on the application than fixed point types. Just saying.

From: Trescott <Trescott@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 09:46:55 +0000

Fixed-point types were primarily for providing fast real operations for real-time systems on processors without hardware floating-point support around 1980. Such types were common on such systems (represented by scaled integers), but were typically re-implemented for each type.

While using Fixed-point can solve a technical issue of not having floating-point hardware, it is not the only reason for using it.

Fixed-point and Floating-point have different Pros/Cons. In many (all?) countries, it is illegal to use floating-point for representing money in software that deals with real money. It is recommended to avoid using floating-point for representing time in games.

So, what's wrong with using floating-point for everything?

As the floating-point … floats around, the size of the step between adjacent values increases or decreases. And that is not acceptable in some use cases. Imagine a bank that used floating-point variables to represent money.

```
type money is digit 6; -- This won't actually
cause the problem shown because the
compiler is free to use more precision than
this. This is only a minimum.
```

If you have a balance of $3,000.00 and you deposit your paycheck of $1,560.32 then your new balance would be $4,560.32. Everything is fine so far.

But what if you are saving up for a down payment on a house, so you are not spending your money as fast as you can.

Now you have a balance of $10,000.00 and you deposit your paycheck of $1,560.32. Now you have 11,560.30! What happened to the 2 cents?!?!

Let's look at how it is represented as a float of 6 digits:

```
  1.00000 x 10^4   | 10,000.0
+ 1.56032 x 10^3   |  1,560.32
---------------------  / ------------
= 1.15603 x 10^4   | 11,560.3
```

Well, you ran out of precision and the cents dropped off.

If you use a fixed point with a step of 0.01 then this would never happen.

```
type money is delta 0.01 range -9_999.99 .. 9_999.99 with Small => 0.01;
```

Now our precision is fixed to the penny. This limits our range but we won't quietly drop your money.

There is a similar problem with games that use a float to store the game time. Often games intend to track time down to fractions of a second. If the game only lasts a few hours it doesn't really matter which is used to store the time. But if the game runs for weeks or months or even years, weird problems may crop up in the game because the programmers assumed a game time resolution in some fraction of seconds but the precision of the timer has been pushed out of the fractions of seconds up to the seconds or tens of seconds.

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 10:44:02 +0000

> There is a similar problem with games that use a float to store the game time [...]

I seem to remember the same is true of the Patriot missile system.

From: JC001 <JC001@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 11:33:21 +0000

> it is not the only reason for using it

I guess I didn't express myself well. One of the HOL project's main targets in the late 1970s was embedded, real-time software. One of the primary reasons that Ada included fixed-point types was their common use in such software.

From: Trescott <Trescott@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 11:43:10 +0000

@JC001 I was just looking up the Ada 83 rationale for this. There is consideration for limitations of the hardware given in the rational, similar to what you said.

See Ada '83 Rationale, Sec 5.1: Introduction (to Ch 5: Numeric Types) at http://archive.adaic.com/standards/83rat/html/ratl-05-01.html#5.1.2

From: sidisyom <sidisyom@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 16:00:11 +0000

Out of curiosity, does anyone have any input on how fixed point types are actually implemented (i.e. software or mapped directly to hardware)? I'm wondering if there's any performance penalty for using these.

If I were to guess I'd say these are implemented in software? I'm thinking, for the IEE754 hardware types the gap increases as the exponent increases (in the normalised range) but remains fixed within a specific exponent so in order to achieve fixed gaps something has to give? Or would the compiler simply reject the type if that is not feasible in hardware (i.e. large numbers, small 'Small's)

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 16:20:04 +0000

If I correctly remember IBM had decimal packed arithmetic supported by the hardware. You could possibly micro-code PDP to extend its instruction set.

As for implementation of binary fixed-point arithmetic using integers it is straightforward, because the representation X = I * F. Addition and subtraction remain as is. Truncating multiplication requires extended multiplication plus a reduction step (shift).

From: JC001 <JC001@forum.ada-lang.io>
Date: Mon, 10 Mar 2025 09:41:46 +0000

> does anyone have any input on how-fixed point types are actually implemented

Fixed-point types are typically implemented as an integer. The value is the integer multiplied by the small. This is why the Money example gives the wrong answer: Value ends up storing 5, which multiplied by 1/128 gives 0.0390625. Rounding to the nearest multiple of the delta results in the output of 0.04.

From: MarioBlunk <MarioBlunk@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 18:10:47 +0000

First, thank you all for joining the discussion.

JC001, you posted an example where the problem was already known back in 1984. So why has the user type the important

```
for Money'Small use 0.01;
```

statement at all? Why is it not default?

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 20:19:50 +0000

Because the default is the most efficient and /correct/ way to implement it.

The program provided by the malicious instructor is semantically wrong. You want a /decimal/ fixed-point number for the currency computations. The program has a /binary/ one, you would rather use in measurements. See RM 3.5.9 [1].

The correct type declaration would be:

```
type Money is delta 0.01 digits 5;
-- Decimal, 5 digits, not for you Elon!
```

This one would work as expected and have 0.01 small.

Forcing a power of 10 small on a binary fixed-point number is equivalent to having a decimal one, which the language has out of the box (and would map onto a machine type on IBM 370). Software decimal fixed-point has a little more overhead because reduction would require a division rather than a mere shift.

[1] http://www.ada-auth.org/standards/22rm/html/RM-3-5-9.html

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Sun, 9 Mar 2025 20:44:09 +0000

The '83 Rationale quoted above seems to be 90% of the way there: fixed-point numbers are integer numbers with a scaling factor applied, what difference does it make if the scale is 0.1 or 0.0625? If you're converting for display, none I think.

There is a difference if you're adding two numbers, one with a small of 0.1, the other with a small of 0.0625. But in that case you as the programmer have to decide what scaling to perform with the appropriate type conversion, it's not left up to the compiler.

Dmitry, I don't think an instructor failing to use a decimal fixed point declaration in 1984 could be called malicious.

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Mon, 10 Mar 2025 09:33:49 +0000

> what difference does it make if the scale is 0.1 or 0.0625?

Let you multiply two binary fixed-point numbers on a binary machine. 0.0625 = 1/32 = 1/2**5 is a power of 2.

```
  0.9375 * 1.875 =
= 30 / 2**5 + 60 / 2**5 =
= 180 / 2**5 / 2**5 =
= 5 / 2**5 = 0.15625
```

Multiplication requires shift by 5 bits.

```
Shift_Right (X * Y, 5)
```

The scale 0.1 is not a power of two and you have to divide by 10. This is a big difference.

Decimal fixed-point can be mapped onto the hardware like this [1]. Some machines had decimal arithmetic. If I remember correctly VAX-11 had decimal strings so DEC Ada 83 compiler could use them for decimal fixed-point. Division by 10 would be a simple shift/slicing. Did DEC Ada that? I do not know.

> I don't think an instructor failing to use a decimal fixed point declaration in 1984 could be called malicious.

I was a joke. In 80s we were allowed to do a lot of things forbidden now. [This is a half-joke]

[1] http://www.simotime.com/asmins01.htm#MP

Ada.Calendar.Formatting.Image

From: reinert <reinert@forum.ada-lang.io>
Subject: Question on Ada.Calendar.Formatting.Image
Date: Tue, 11 Mar 2025 21:12:40 +0000
Newsgroups: forum.ada-lang.io

Ada.Calendar.Formatting.Image(10_000) gives the string "02:46:40", but it seems my compiler does not accept durations of 100 hours or more. I can create a workaround, but am I missing something?

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Tue, 11 Mar 2025 21:53:15 +0000

The ISO 8601-1:2019 mandates [T]hh:mm:ss, so the upper limit is 100 hours - 1s. Ada RM says [1] that if it is more than 100 hours the result is implementation defined.

[1] http://www.ada-auth.org/standards/22rm/html/RM-9-6-1.html

Fully Specifying Packages

From: sbenitezb <sbenitezb@forum.ada-lang.io>
Subject: Fully specifying packages
Date: Fri, 14 Mar 2025 00:03:59 +0000
Newsgroups: forum.ada-lang.io

What has been your experience fully specifying multiple packages before implementing the bodies, either in full Ada or with SPARK?

I'm thinking of giving it a try for my current project if it gives good results.

From: Micronian2 <Micronian2@forum.ada-lang.io>
Date: Fri, 14 Mar 2025 03:22:08 +0000

I've found it useful to have multiple package specs defined first and write example programs that utilize some of them. This usually helps to identify details or interactions I didn't realize at first which lead me to make adjustments. This is why I really like Ada's emphasis on separation of specification and implementation, which helps to discourage the author from getting too caught up in implementation details too early only to find out in the middle that something has to be reworked.

From: sttaft <sttaft@forum.ada-lang.io>
Date: Fri, 14 Mar 2025 14:13:41 +0000

I am personally a big fan of "top-down design" which is pretty close to what you are suggesting. There might be a bit less re-use using a top-down development process, though that can typically be ameliorated a bit during the lower-level implementation phase. In my experience, the net result of a top-down design is a significantly better overall architecture, and in some cases, that results in /more/ reusable components when you are done.

I would definitely agree with writing as many example programs as you can before locking down an API, because that is really the best way to validate the architecture and usefulness of the set of package specs.

From: jere <jere@forum.ada-lang.io>
Date: Fri, 14 Mar 2025 20:08:57 +0000

I'm thinking of giving it a try for my current project if it gives good results.

I have found it really useful in multiple projects. I did find myself struggling with how to do it when I first started and I would often fall into habits of trying to implement things too early, but once I got used to it, I really liked how the design stage felt, and I felt my designs had much better architecture and were easier to document. I'm saying it is the silver bullet for everyone, but for me it feels really good.

> There might be a bit less re-use using a top-down development process

From my own experience I just found it to be a different type of reuse. Going top down on my designs allowed me to reuse entire systems, only changing a few implementations at the low level while the rest of the code just worked. For the projects doing bottom up, I ended up with components that I could reuse in multiple different types of projects but I had to design the business logic from scratch in most cases.

From: JC001 <JC001@forum.ada-lang.io>
Date: Sat, 15 Mar 2025 11:33:40 +0000

This is pretty much the only way to engineer software. I generally use an edges-in approach, which results in both low-level reusable packages (for hardware interfacing, for example) and high-level package specifications representing problem-space entities. One then repeats the process to implement the bodies of those high-level packages, and the bodies of the packages resulting from implementing those bodies, and so on, until things get low-level enough to be easily implemented correctly.

Disable Secondary Stack Completely

From: jgrivera67 <jgrivera67@forum.ada-lang.io>
Subject: How can I tell GNAT to disable the secondary stack completely on bare-metal Ada code?
Date: Fri, 14 Mar 2025 12:48:12 +0000
Newsgroups: forum.ada-lang.io

t looks like pragma Restrictions (No_Secondary_Stack); does not work for the environment task, according to the documentation fragment below:

3.2.3. Disabling the Secondary Stack [1]

The secondary stack can be disabled by using pragma Restrictions (No_Secondary_Stack); This will cause an error to be raised at compile time for each call to a function that returns an object of unconstrained type. When this restriction is in effect, Ada tasks (excluding the environment task) will not have their secondary stacks allocated.

Does anybody know how I can disable the secondary stack completely, so that not even the environment task can use it?

[1] https://docs.adacore.com/gnat_ugx-docs/html/gnat_ugx/gnat_ugx/the_stacks.html#disabling-the-secondary-stack

From: jere <jere@forum.ada-lang.io>
Date: Fri, 14 Mar 2025 13:59:21 +0000

Are you making the runtime or using an existing runtime?

Some untested thoughts:

If you are making it yourself (or have access to recompiling the runtime you are using), you just leave out the files that define the stack (I think s-secsta.ads/b ?). If you are using an existing runtime, it might be tougher. You could see if the linker script defines them and if so remove those definitions, which maybe could cause a linking error if you tried to do anything that requires secondary stack (that includes using tagged types since they return unconstrained objects internally sometimes).

From: simonjwright <simonjwright@forum.ada-lang.io>
Date: Fri, 14 Mar 2025 14:59:24 +0000

Part of the trouble is that when the binder generates the main program/environment task it includes secondary stack code, and you don't want to have to build your own gnatbind.

From: jgrivera67 <jgrivera67@forum.ada-lang.io>
Date: Fri, 14 Mar 2025 18:25:14 +0000

Thanks for the response. I'm using my own bare-minimum runtime. I already removed the s-secsta.ads and s-secsta.adb from my runtime. So, I can catch at compile-time any unwanted calls that return an unconstrained type and thus would cause the use of the secondary stack, and therefore would generate a compiler error. However, I'm hitting runtime crashes on calls like the following:

```
Str : String (1 .. 8);
Fill_Str (Str);
Foo ("XXXX" & Str);
```

Where, Foo has the following signature:

```
procedure Foo (S : String);
```

So, it seems that using the "XXXX" & Str expression as a parameter to Foo is placing the resulting string on an uninitialized secondary stack, which corrupts the primary stack?

From: jere <jere@forum.ada-lang.io>
Date: Fri, 14 Mar 2025 20:01:41 +0000

That's interesting, I would have expected it to fail at link time since the secondary stack is needed and it doesn't exist.

Free Monads in Ada

From: gast <gast@forum.ada-lang.io>
Subject: Free monads in Ada
Date: Tue, 25 Mar 2025 20:36:30 +0000
Newsgroups: forum.ada-lang.io

Hi, I just wanted to ask if there is anyone who has implemented free monads ([1]) in Ada. I'm not sure if it's feasible in Ada because it uses FP constructs which are probably not so easy to represent in Ada.

[1] https://github.com/gbogard/free-monads-from-scratch

From: OneWingedShark <OneWingedShark@forum.ada-lang.io>
Date: Tue, 25 Mar 2025 22:37:46 +0000

Try:

- Functional Programming in…Ada? [1]
- Full functional programming in a declarative Ada dialect [2]
- From ML to Ada: Strongly-typed language interoperability via source translation [3]
- Exploring the boundaries of Ada syntax with functional-style iterators [4]

[1] https://okasaki.blogspot.com/2008/07/functional-programming-inada.html

[2] https://www.academia.edu/80710117/Full_functional_programming_in_a_declarative_Ada_dialect

[3] https://www.cambridge.org/core/services/aop-cambridge-core/content/view/9ACB018A7F71ECF8A0FEE0DFEDAC4E84/S0956796898003086a.pdf/div-class-title-from-ml-to-ada-strongly-typed-language-interoperability-via-source-translation-div.pdf

[4] https://www.researchgate.net/profile/Alejandro-Mosteo/publication/349330900_Exploring_the_boundaries_of_Ada_syntax_with_functional-style_iterators/links/603ce4654585154e8c6b8641/Exploring-the-boundaries-of-Ada-syntax-with-functional-style-iterators.pdf

From: pmnw <pmnw@forum.ada-lang.io>
Date: Wed, 26 Mar 2025 21:59:20 +0000

> Exploring the boundaries of Ada syntax with functional-style iterators

This is wonderful. I like it so much I would nominate the author of this paper to be Ada's BDFL. Of all the ways to program, I miss the functional style and higher-order functions the most in Ada.

From: OneWingedShark <OneWingedShark@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 02:27:17 +0000

I am slightly disappointed they didn't mention the Proposal for solving the issue (Abstracting TYPE and INTERFACE) [1] that I put together — note that there is no syntax proposed because I want the notion ironed-out before bikeshedding; but the idea is that there are two things we need for Ada:

- A method to "turn the ability (ala generics) to statically construct types by parts and properties inside-out", thus allowing us to "hang" proofs on an abstraction, then marry/integrate that into the type (completely absent tagged-typing/OOP).
- A method to describe the "how" of interfacing/interacting a type, complete with attributes and properties, thus allowing a generalization of things like indexing. (e.g. an access's auto-dereferencing is /exactly/ indexing by a zero-length list of indices; completely consistent with how Ada would reduce "function K() return data" to function k return data.)

These two features would allow for the materialization of the notional type-hierarchy, as well as clearing most of the clutter of SPARK-proof and/or user-defined-indexing… in fact, given implementing the abstraction-on-interface (of which indexing is a subcomponent), it would allow us to drop the currently-needed auxiliary-types used in user-indexing.

[1] https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues/104

From: pmnw <pmnw@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 09:52:36 +0000

> the Proposal for solving the issue

Judging by Lucretia's input on the proposal page, this reduces to having a way to construct (or even compose) types by certain traits, correct?

You'd want to define a set of type traits (constraints, really) and then compose your type hierarchy from those traits and of course use them as formal parameters to functions, for generics, and proofs. This is somewhat achievable with contracts but contracts are not objects - they cannot be packed behind an identifier and reused, can they? They're not interfaces, just annotations.

This idea has been implemented rigorously in Haskell's type classes and Rust's type traits and I have exclusively heard praise for those. I think they have particular appeal for those with mathematical disposition.

To pair that with Ada's already phenomenal type system would be wonderful, but this would be an advanced feature, wouldn't it?

I am biased but I think importing some of the functional idioms (a general tendency in language design nowadays that's actually good and sensible because it can be used to reduce complexity) would be more desirable from the perspective of an average engineer.

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 12:11:16 +0000

> This is somewhat achievable with contracts but contracts are not objects - they cannot be packed behind an identifier and reused, can they?

Why on earth contracts need to be objects, unless within the compiler?

Of course, contracts can be reused without making them run-time objects. Interface inheritance is the vehicle for that.

> They're not interfaces, just annotations.

Interface is a contract an implementation must fulfill.

Contract is not the program and shall have no run-time effect, whatsoever. Conflating the meta language of contracts with the object language of the program is a road to nowhere.

> importing some of the functional idioms would be more desirable from the perspective of an average engineer

Ada must stay purely imperative language, IMO.

From: pmnw <pmnw@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 13:45:14 +0000

> Why on earth contracts need to be objects, unless within the compiler?

I am not thinking of type traits occupying space in memory or about constructing them and passing them around to functions as arguments.

Perhaps I shouldn't have mentioned "objects" in my message above. Sorry for the confusion.

On the contrary, I'm thinking solely about composing types by including their capabilities in the program text and that being completely static with no runtime effect whatsoever.

If such type composability can be done with Ada interfaces, I admit that's new to me.

> Ada must stay purely imperative language, IMO.

This seems to be the consensus judging by the general lack of functional influences in Ada.

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 17:00:33 +0000

> On the contrary, I'm thinking solely about composing types by including their capabilities in the program text and that being completely static with no runtime effect whatsoever.

So, it is about types algebra. Then I do not see how this is related to functional programming which basically strives to remain un/ad-hoc-typed.

If you decompose the problem into functions, you naturally throw types out. Ada programming decomposes the problem into abstract user-defined types. Functions are secondary as they apply to types. Functional and relational hold types secondary. You cannot have both. These are different paradigms for a reason. Poisoning Ada with functional mess would produce nothing but more mess.

> If such type composability can be done with Ada interfaces, I admit that's new to me.

Inheritance/derivation is the basic type algebraic operation. The way must be in my view to replace as much of built-in types magical operations like:

type T **is array** (…) **range** …;

with universal inheritance, e.g. in this case inheritance from some array interface.

From: pmnw <pmnw@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 19:25:42 +0000

> So, it is about types algebra. Then I do not see how this is related to functional programming which basically strives to remain un/ad-hoc-typed.

It is not bound to FP - it was the Mosteo & Lorente (2021) paper that was about the functional-style operations on containers.

The topic of types came from the message by @OneWingedShark, or rather from my cursory reading of it and the related proposal on Github.

> If you decompose the problem into functions, you naturally throw types out.

Well, ML and Haskell are statically typed and allow type annotations. Haskell has type traits. I'm not sure if this can be seen as throwing the types out.

> Poisoning Ada with functional mess would produce nothing but more mess.

I think you're right - Ada is an imperative language and it was not designed for functional programming as exhibited in the paper (those right-hand side examples of filter/take/map/fold/etc.).

> The way must be in my view to replace as much of built-in types magical operations [...] with universal inheritance

This may be a lot to ask, but do you have any code that does this thing that a relative newcomer like me could read and learn from? I want to see how it's done in idiomatic Ada.

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 19:59:48 +0000

> This may be a lot to ask, but do you have any code that does this thing that a relative newcomer like me could read and learn from? I want to see how it's done in idiomatic Ada.

Ada cannot do this. You cannot have a class of array types. As I said operations producing arrays, records, access, scalar task types were hard-wired in Ada 83. Ada 95 added tagged types orthogonal to anything else. They use a very limited record extension model unsuitable for scalar types.

After that various hacks were applied to avoid solving pressing type system problems. The container library, unbounded strings, protected objects all fall out of the type system. E.g., neither unbounded string nor vectors are arrays.

Parallel to that, generics were intensively misused to create some resemblance of classes, but since generics are static, they lack run-time instances, which produces massive overdesign and incredible kludges.

Then there were helper types in the worst C++ fashion.

Now it seems functional stuff, just another hack.

Tomorrow it will be something else.

From: pmnw <pmnw@forum.ada-lang.io>
Date: Sat, 29 Mar 2025 00:00:30 +0000

If there's one thing I learned from this thread so far is that there are differing opinions about what could be added or changed in Ada, but all of you want the language to improve.

> After that various hacks were applied to avoid solving pressing type system problems. […] Now it seems functional stuff, just another hack. […] Tomorrow it will be something else.

Alright, I see your point. I cannot judge it because I don't know Ada as well as you do, but it sounds plausible to me.

Basically, to my "it would be nice to have some convenience of functional programming", you're saying "no, you don't know the kind of mess that's underneath".

Likely couldn't have been avoided, given how from Ada83 perspective even OOP is a foreign influence.

[...]

From: OneWingedShark <OneWingedShark@forum.ada-lang.io>
Date: Sat, 29 Mar 2025 04:00:45 +0000

> Ada already gives a good first impression - the type system is /so good/ - but then you get to the "exposition dump" of OOP, packages and generics. I think it's telling that new languages seem to stay away from supporting OOP and generics at the same time.

Ada did generics the right way: they are *not* textual-substitution templates, but actual constructs /*and*/ the formal parameters are the exposing of the type-system-properties to the body of the generic. (Some of the particular syntax in generic formal-parameters is a bit messy, but it's relatively minor amounts.) — as an example consider Type X() is limited private; and Type Y is (); and type Z() is new Parent;: these are, respectively, "any-type", a discrete/enumeration type, and a type derived from parent.

As you can see, this presents to the implementation the information that it can rely on, while at the same time presenting to the instantiation the requirements for the parameters, too. — And Z (OOP) is a relative non-issue for generics.

> I'm going to try to wrap my head around your wider message to the extent my still-too-limited Ada knowledge allows, but the allure of this part is hard to dismiss. I imagine looking at such code would make prospective users realise the amazing things you can do, and once you know the type system, you can get so much done.

Right, the type system is excellent. I believe the generic system is excellent as well, though there are a few things that would make it "more usable", and @dmitry-kazakov would argue that generics are not good due to some previous generic-heavy code and GNAT's method of handling them.

There's the "Three Papers" that I wrote /Explaining Ada's Features/, which may help you:

- *Explaining Ada's Packages [1]*,
- *Explaining Ada's Object Oriented Programming [2]*, and
- *Explaining Ada's Generics [3]*.

Let me know if you find them helpful.

> Yet, going back to earth, I can't say if these are doable or feasible because the language is what it is. There's no reason for me to believe that the custodians of Ada don't have its best interest in mind, but the "fix past mistake, remove bloat, add cool features" button has not yet been invented as far as I know.

I am on the ARG, and IMO the user-defined indexing is a mistake as evidenced by how awkward it is to implement, requiring helper-types and otherwise extraneous code cluttering everything up — that's why I really want to do the abstract-the-interface and /Annex J/ (Obsolescent features) the current aspect-based user-indexing system.

SPARK proof is an amazing tool, and I'm quite impressed with it, but a lot of SPARK proven code has a LOT of clutter for the various aspects needed for the prover; and a LOT of these are duplicated-in-large-parts for something like a CONTAINERS library or, indeed any library which has a collection of substantially-similar abstract-data-types. Hence why I want the second feature of abstracting the type-properties out (in a way that is not necessarily/connected-to OOP).

> I expect it could feel similar if Ada, but then again, it also wasn't designed with OOP in mind and it didn't prevent it from following the market in 1995. How well it served Ada is up for discussion.

I highly recommend you take a look at the OOP paper, definitely reread the Packages paper first, and /then/ I'm sure you'll be a bit more reserved in saying Ada wasn't designed with OOP in mind: the four pillars of OOP were already present in Ada83, but as orthogonal features.

[1] https://web.archive.org/web/20230930004246/http://edward.fish/wp-content/uploads/2023/07/Explaining-Ada%E2%80%99s-Packages.pdf

[2] https://web.archive.org/web/20230930004514/http://edward.fish/wp-content/uploads/2023/07/Explaining-Ada%E2%80%99s-OOP.pdf

[3] https://web.archive.org/web/20230930004650/http://edward.fish/wp-content/uploads/2023/07/Explaining-Ada%E2%80%99s-Generics-1.pdf

From: pmnw <pmnw@forum.ada-lang.io>
Date: Sat, 29 Mar 2025 09:24:24 +0000

> There's the "Three Papers" that I wrote /Explaining Ada's Features/, which may help you.

Thank you. I had no idea these existed, that's very helpful. I prefer this form of presentation to what's offered on AdaCore's Learn platform which had too low information density and annoyed me with its fragmentation when I last tried it.

> Ada did generics the right way

Yes, like many things in Ada, their design was literally ahead of its time. For instance, after Alex Stepanov and Dave Musser failed to build their generic library in Ada, Stepanov moved on to C++ and tried to convince Bjarne Stroustrup to base C++ templates on Ada generics. [1]

> Alex then worked at HP Labs but he had earlier worked for a couple of years at Bell Labs, where he had been close to Andrew Koenig and where I had discussed library design and template mechanisms with him. He had inspired me to work harder on generality and efficiency of some of the template mechanisms, but fortunately he failed to convince me to make templates more like Ada generics. Had he succeeded, he wouldn't have been able to design and implement the STL! – Bjarne Stroustrup [1] 4.1.2 The STL emerges

> The Scheme work led to a grant to produce a generic library in Ada. Dave Musser and I produced a generic library that dealt with linked structures. My attempts to implement algorithms that work on any sequential structure (both lists and arrays) failed because of the state of Ada compilers at the time. I had equivalences to many STL algorithms, but could not compile them. Based on this work, Dave Musser and I published a paper where we introduced the notion of generic programming insisting on deriving abstraction from useful efficient algorithms. The most important thing I learned from Ada was the value of static typing as a design tool. Bjarne Stroustrup had learned the same lesson from Simula. – Alex Stepanov [1] 4.1.8 Stepanov's view

Please note also the mention of Simula. I remember that Ada 83 Rationale mentions it too and that works in favour of your claim that OOP was very much on the horizon.

> IMO the user-defined indexing is a mistake

Hence why I want the second feature of abstracting the type-properties out (in a way that is not necessarily/connected-to OOP).

Is the argument here that some of these advanced features have been implemented using "hacks" on top of what was available (such as being tied to OOP facilities) and not as general features of the language? The effect is they're complicated, impure and inelegant, right? I'm lenient on that because that's 40 years of nature at work, but I agree that such things are confusing. For example, I was scratching my head when I read about the Ada 2022 reduction expressions – A'Reduce ("+", 0) (an FP influence btw) – because this I would expect to simply be a higher-order function in the standard library. From the outside, it looks like some kind of function call syntax, and you wonder what else – like Map/Transform/Filter/My_Arbitrary_Func – can be put there instead. It turns out it's just a singular thing - not very cool! I'm sure there are reasons for that and it's a useful thing to have, but it's confusing.

> If you want to see what FP in Ada could look like, there is Oberon+ which has extended Oberon a lot and includes FP, it looks weird though.

Well, my primary consideration would always be the reduction of complexity and ease-of-expression which improves readability and comprehension. These aims can only be achieved without introducing foreign syntax or complicating the existing one (like multiplying ways to do things!). Secondary perhaps is compiler and implementation hygiene.

So my stance from the beginning is that many functional idioms are a /strong drug/, but if they were to pollute a language, they're not worth it.

[1] Bjarne Stroustrup - Evolving a language in and for the real world

https://www.stroustrup.com/hopl-almost-final.pdf

From: dmitry-kazakov <dmitry-kazakov@forum.ada-lang.io>
Date: Sat, 29 Mar 2025 09:30:16 +0000

> it also wasn't designed with OOP in mind and it didn't prevent it from following the market in 1995.

It was. Ada was object-based. The type system supported derivation and inheritance but lacked run-time class-wide objects. So generic programming was not possible. GP means "programming in terms of sets of types."

In Ada 83 GP was achieved per generics, which then seemed a good compromise because macros were fashionable and considered a good idea. C had a preprocessor, PL/1 had macros, so Ada had got its own macro language of generics. It was even weakly typed, a nice contrast to common untyped macros…

> Right, the type system is excellent. I believe the generic system is excellent

> as well, though there are a few things that would make it "more usable", and @dmitry-kazakov would argue that generics are not good due to some previous generic-heavy code and GNAT's method of handling them.

No, I would argue that static parametric polymorphism is fundamentally flawed:

- It constitutes a meta-language, generic type is not a type, generic package is not a package etc. It is not Ada by definition.
- Its classes have no-run-time instances. You cannot have a class-wide object holding values from a set of generic instances.

Create a generic widget library and see how it works. It does not. Parametric polymorphism inherently cannot do that.

> I'm unconvinced about the added value. I like a lot about it, but it's a mere shadow of the /real thing/ in languages that weren't designed with FP inventions.

True. The /real thing/ is a lie. The sole purpose of any program is the /side effects/ of its execution. But that goes quickly over the board and what remains is harmful programming practice which can be seen in the languages striving to appear functional (no pun intended … or maybe).

The problem is wide vs narrow approach. Wide operations apply to large objects as a whole. That is how FP decomposition works. You do something to all container, to whole string etc. This is how tokenizing and other mess comes into play. Take container, get a new one. To extract an element of a matrix, multiply it by two vectors. Exciting!

This is far worse than infamous OOP orgies of creating countless meaningless classes. From the software design and engineering decomposition into wide operations is an /extremely/ bad idea for obvious reasons. And this is the very first thing every newbie or lazy programmer grasps and never lets go.

From: OneWingedShark <OneWingedShark@forum.ada-lang.io>
Date: Thu, 27 Mar 2025 20:16:01 +0000

> this reduces to having a way to construct (or even compose) types by certain traits, correct?

I've not really seen a good example of a "traits system" in my programming-language research; given the intuitive leap in how it's described, it could be exactly what's being proposed, but I hesitate because, as I said, I don't have any real experience on a "traits system". (Things like e.g. C++ concepts are claimed to be equivalent to Ada's contracts+generics, it seemed like a pile of confusing and incoherent mess to me when I read up on the proposed feature.)

> You'd want to define a set of type traits (constraints, really) and then compose your type hierarchy from those traits and of course use them as formal parameters to functions, for generics, and proofs. This is somewhat achievable with contracts but contracts are not objects - they cannot be packed behind an identifier and reused, can they? They're not interfaces, just annotations.

There's two features in view here: one is essentially the computer-science /Abstract Data Type/, where you can lay out the properties, perhaps some [sub]structure, and from this construct hang the proofs/annotations for SPARK — this is to (a) clean up SPARK code and improve its readability, and (b) provide a form of "parameterization" and "early+late-binding" of proofs on the code [because you can separate into two the proof onto the abstraction, and the implementation of the abstraction therefore fulfils the proof]. — But they don't have to be "objects" at all, in the OOP-sense; they would have to be an "object" in the compiler/language sense (like a variable is).

The other feature being the abstraction of the interface, the "how this is handled/interacted-with in the language", of the type, things like indexing, dereferencing, attributes, properties. Indexing is the most obvious example, but also things like dereferencing, and (things like) limited/non-limited and constrained/unconstrained. (Honestly, being able to reify the conceptual-type hexarchy in that picture would go a *LONG* way to making Ada more friendly for newcomers: /"Just look in Ada.Meta.Type_System, you'll see all the various classes of types, and their attributes."/)

> This idea has been implemented rigorously in Haskell's type classes and Rust's type traits and I have exclusively heard praise for those. I think they have particular appeal for those with mathematical disposition.

Indeed-so.

It's why I was particularly disgusted with the proposed/GNAT-experimental class-syntax — IMO, it's a stupid, "/let's be more like python and C++!/" feature… when we could literally be solving the entire problem, unifying, and simplifying the language. (One of the benefits of the abstraction-of-interfaces/type-classes/whatever is that you can then use that to define the types, and model-check/prove them, hoisting the LRM definitions into the LRM's own constructs. [i.e. allowing SPARK-style proof on metasystem, and therefore on the type-system].)

> I am biased but I think importing some of the functional idioms (a general tendency in language design nowadays that's actually good and sensible because it can be used to reduce complexity) would be more desirable from the perspective of an average engineer.

Exactly the reason I want to do so, "on a meta-level"… we can then use the language to define the language; which (done correctly) will make things easier both for compiler-writers and language-users/-learners.

> Ada must stay purely imperative language, IMO.

I have mixed feelings there: I would be in agreement if we were to import enough "functional" to allow subprograms-as-parameters, without having [typically anonymous] access types — in fact, there would be a *LOT* of clean-up if anonymous access were removed altogether.

> This seems to be the consensus judging by the general lack of functional influences in Ada.

I think that with a /*LOT*/ of what you would want functional-programming for, you could get with the abstract-interface I proposed and/or generics. The first link in my first reply shows some of the difficulties, though it also seems unaware of generics.

> If you decompose the problem into functions, you naturally throw types out. Ada programming decomposes the problem into abstract user-defined types.

Not necessarily; check out "How to Think about Parallel Programming: Not!" [1] — pay particular attention to the properties mentioned at 57:44, and also to the split-string example at about 40:00.

[1] https://www.infoq.com/presentations/Thinking-Parallel-Programming/

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

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

April 05 **Ada Monthly Meetup 2025 March**, Internet. New edition of the monthly online meeting to gather the Ada community, see each other, talk about some things, and let people present or showcase their work and discuss the news.

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

April 07-10 31st **International Working Conference on Requirements Engineering: Foundation for Software Quality** (REFSQ'2025), Barcelona, Spain. Theme: "Social REsponsibility".

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

Apr 27 – May 03 47th **International Conference on Software Engineering** (ICSE'2025), Ottawa, Ontario, Canada. Topics include: the full spectrum of Software Engineering (SE), trustworthy AI for SE; AI-assisted software design and model driven engineering; mining software repositories; software metrics (and measurements); software design methodologies, principles, and strategies; architecture quality attributes, such as security, privacy, performance, reliability; modularity and reusability; dependency and complexity analysis; patterns and anti-patterns; technical debt in design and architecture; formal methods and model checking; reliability, availability, and safety; resilience and antifragility; design for dependability and security; vulnerability detection to enhance software security; dependability and security for embedded and cyber-physical systems; evolution and maintenance; API design and evolution; software reuse; refactoring and program differencing; program comprehension; reverse engineering; environments and software development tools; human and social aspects (focusing on programming languages, environments, and tools supporting individuals, teams, communities, and companies; focusing on software development processes; ...); modeling and model-driven engineering; variability and product lines; modeling languages, techniques, and tools; empirical studies on the application of model-based engineering; software testing; automated test generation techniques such as fuzzing, search-based

approaches, and symbolic execution; testing and analysis of non-functional properties; program analysis; debugging and fault localization; runtime analysis and/or error recovery; etc.

April 27-28 13th **International Conference on Formal Methods in Software Engineering** (FormaliSE'2025). Topics include: approaches, methods and tools for verification and validation; formal approaches to safety and security related issues; scalability of formal method applications; integration of formal methods within the software development lifecycle; model-based engineering approaches; correctness-by-construction approaches for software and systems engineering; application of formal methods to specific domains, e.g., autonomous, cyber-physical, intelligent, and IoT systems; formal methods in a certification context; case studies developed/analyzed with formal approaches; experience reports on the application of formal methods to real-world problems; guidelines to use formal methods in practice; usability of formal methods; etc.

Apr 28-29 37th **International Conference on Software Engineering Education and Training** (CSEET'2025), Ottawa, Ontario, Canada. Topics include: all dimensions of learning and teaching in the area of software engineering.

May 03-08 28th ETAPS **International Joint Conferences on Theory and Practice of Software** (ETAPS'2025), Hamilton, Canada.

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

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

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

May 07-08 31st **International Symposium on Model Checking of Software** (SPIN'2025). Topics include: automated tool-based techniques for the analysis of software as well as models of software, for the purpose of verification and validation. Deadline for submissions: April 7, 2025 (non-tool artifacts).

May 06-09 18th **Cyber-Physical Systems and Internet of Things Week** (CPS-IoT Week'2025), Irvine, USA. Event includes: 5 top conferences, HSCC, ICCPS, IoTDI, IPSN, and RTAS, as well as poster and demo sessions, workshops, tutorials, competitions, industrial exhibitions, PhD forums, and summits. Deadline for submissions: April 11, 2025 (student travel award applications). Deadline for early registration: April 14, 2025.

☺ May 06-09 31st IEEE **Real-Time and Embedded Technology and Applications Symposium** (RTAS'2025). Topics include: applications with timing requirements; real-time and embedded operating systems; application profiling, WCET analysis, compilers, tools, benchmarks and case studies; modelling languages, modelling methods, model learning, model validation and calibration; scheduling and resource allocation; schedulability and response time analyses; verification and validation methodologies; etc. Deadline for early registration: April 2, 2025.

May 06-09 16th ACM/IEEE **International Conference on Cyber-Physical Systems** (ICCPS'2025). Topics include: safety and resilience for CPS; software platforms and systems for CPS; specification languages and requirements; design, optimization, and synthesis; testing, verification, certification, assurance; security, trust, and privacy in CPS; tools, testbeds, demonstrations and deployments; CPS applications in power systems, infrastructure networks, transportation, healthcare, automotive, aerospace, etc. Deadline for early registration: April 2, 2025.

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

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

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

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

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

June 1 **Ada-Belgium Spring 2025 Event,** Leuven, Belgium. Includes: 2025 Ada-Belgium General Assembly and Ada Round-Table Discussion. Deadline for registration: May 28, 2025, 18:00.

☺ June 02-06 **International Conference on the Art, Science, and Engineering of Programming** (Programming'2025), Prague, Czech Republic.

June 2 9th **Workshop on Modern Language Runtimes and Ecosystems** (MoreVMS'2025). Topics include: design, implementation, and usage of modern languages and runtimes; interoperability between languages, tooling support (e.g. debugging, profiling, etc.), programming language development environments, case studies of existing language implementation approaches, language implementation challenges and trade-offs, surveys and usage reports to understand usage in the wild, ideas for how we should build languages in the future, etc. Deadline for early registration: April 18, 2025.

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

June 03 26th IEEE **International Workshop on Parallel and Distributed Scientific and Engineering Computing** (PDSEC-2025). Topics include: methodologies and

experiences used in scientific and engineering applications and algorithms to achieve sustainable code development for better productivity, application performance and reliability; languages for scientific computing on hybrid systems; tools and techniques for improving the performance, reliability and resilience of scientific applications; etc.

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

June 13 AEiC'2025 - 2nd **Ada Developers Workshop**. Topics include: everything related to Ada software development, i.e. similar to last year's Ada Developers Workshop and to the Ada DevRooms at FOSDEM, technical presentations, tutorials, demos, live performances, project status reports, discussions, etc., offering a place where the Ada community can meet and share their work and projects. Deadline for submissions: April 6, 2025 (round 1), open (round 2). Registration: low-cost in-person fee & free remote participation.

June 13 AEiC'2025 - 4th ADEPT workshop, AADL by its practitioners (ADEPT'2025). Topics include: current projects in the field of design, implementation and verification of critical systems where AADL is a first-citizen technology. Deadline for submissions: April 30, 2025 (abstracts).

June 10-13 **Software Technologies: Applications and Foundations** (STAF'2025), Koblenz, Germany. Topics include: practical and foundational advances in software technology. Deadline for early registration: May 9, 2025.

June 12-13 18th ACM SIGPLAN **International Conference on Software Language Engineering** (SLE'2025). Topics include: software language engineering in general, rather than engineering a specific software language, such as software language design and implementation, software language validation (verification and formal methods for languages, testing techniques for languages, simulation techniques for languages, ...), software language integration and composition, software language maintenance (software language reuse; language evolution; language families and variability, language and software product lines), domain-specific approaches for any aspects of SLE (design, implementation, validation, maintenance), empirical evaluation and experience reports of language engineering tools (user studies evaluating usability, performance benchmarks, industrial applications), etc. Deadline for submissions: April 6, 2025 (arteifact evaluation committee nominations), April 23, 2025 (artifacts). Deadline for early registration: May 10, 2025.

☺ June 16-17 PLDI2025 - 26th ACM SIGPLAN/SIGBED **International Conference on Languages, Compilers and Tools of Embedded Systems** (LCTES'2025), Seoul, South Korea. Co-located with PLDI'2025. Topics include: programming language challenges (domain-specific languages; features to exploit multicore architectures; features for distributed and real-time control embedded systems; capabilities for specification, composition, and construction of embedded systems; language features and techniques to enhance reliability, verifiability, and security; compiler challenges; ...), interaction between embedded architectures, operating systems, and compilers (support for enhanced programmer productivity; support for enhanced debugging, profiling, and exception/interrupt handling; optimization for low power/energy, code/data size, and real-time performance; tools for analysis, specification, design, and implementation; hardware, system software, application software, and their interfaces; distributed real-time control; system integration and testing; run-time system support for embedded systems; support for system security and system-level reliability; ...), predictability of resource behavior: energy, space, time (validation and verification, in particular of concurrent and distributed systems; formal foundations of model-based design as the basis for code generation, analysis, and verification; ...), design and implementation of novel architectures (architecture support for new language features, virtualization, compiler techniques, debugging tools; ...), etc. Deadline for submissions: April 28, 2025 (artefacts).

June 16-20 20th **International Federated Conference on Distributed Computing Techniques** (DisCoTec'2025), Lille, France. Includes the COORDINATION, DAIS, and FORTE conferences. Topics include: a broad

spectrum of distributed computing subjects, from theoretical foundations and formal description techniques, testing and verification methods, to language design and system implementation approaches. Deadline for submissions: mid April, 2025 (workshop papers). Deadline for registration: May 23, 2025 (early), June 11, 2025 (late).

June 23-27 — 33rd ACM **International Conference on the Foundations of Software Engineering** (FSE'2025), Trondheim, Norway. Topics include: debugging and fault localization; dependability, safety, and reliability; embedded software, safety-critical systems, and cyber-physical systems; model checking; model-driven engineering; parallel, distributed, and concurrent systems; program analysis; programming languages; software architectures; software engineering education; software evolution; software security; software testing; software traceability; symbolic execution; tools and environments; etc. Deadline for submissions: May 23, 2025 (NSF student travel support applications). Deadline for early registration: April 16, 2025.

June 25-28 — 34th ACM SIGSOFT **International Symposium on Software Testing and Analysis** (ISSTA'2025), Trondheim, Norway. Topics include: theory, design, implementation, optimization, testing, and analysis of software systems, programs and programming languages. Deadline for submissions: April 2, 2025 (travel and childcare grant applications), May 23, 2025 (student travel support applications). Deadline for early registration: April 16, 2025.

Jun 28 – Jul 03 — 25th **International Conference on embedded computer Systems: Architectures, MOdeling and Simulation** (SAMOS'2025), Samos Island, Greece. Topics include: improving performance, power efficiency, reliability, dependability, ..., in embedded systems in fields such as automotive, space, avionics, medical, edge computing, ...; design methodologies and tools for embedded systems; modeling and simulation; compilation techniques; reliability analysis; system-level design; programming models and development tools for embedded systems; profiling, measurement, and performance analysis techniques; compilation, optimization strategies, and code generation; verification, testing, and debugging methodologies; etc. Deadline for submissions: April 22, 2025 (special-session submissions).

☺ Jun 30 – Jul 04 — 39th **European Conference on Object-Oriented Programming** (ECOOP'2025), Bergen, Norway. Topics include: all topics related to programming languages, software development, systems and applications; all practical and theoretical investigations of programming languages, systems and environments; innovative solutions to real problems as well as evaluations of existing solutions. Deadline for registration: May 28, 2025 (early), June 22, 2025 (late).

- ☺ July 02 — 20th **Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems** (ICOOOLPS'2025). Topics include: implementation and optimization of fundamental languages features; abstraction lowering and representation techniques (exceptions, concurrency, ...); runtime systems technology; compiler toolchains (intermediate representations, ...); compiler retargeting; resource-sensitive systems (real-time, low power, mobile, cloud); tooling support, debuggability and observability of languages as well as their implementations; empirical studies on language usage; etc. Deadline for submissions: May 28, 2025 (papers).
- ☺ July 03 — 27th **International Workshop on Formal Techniques for Judicious Programming** (FTfJP'2025). Topics include: language design and semantics, type systems, concurrency and new application domains, specification and verification of program properties, program analysis (static or dynamic), security, pearls (programs or proofs), etc. Deadline for submissions: May 7, 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. Deadline for early registration: June 1, 2025.

July 08-11 — 37th **Euromicro Conference on Real-Time Systems** (ECRTS'2025), Brussels, Belgium. Topics include: all aspects of timing requirements in computer systems; elements of time-sensitive software systems, such as operating systems, hypervisors, middlewares and frameworks, programming languages and compilers,

runtime environments, ...; real-time applications, such as modeling, design, simulation, testing, debugging, and evaluation in domains such as automotive, avionics, control systems, industrial automation, robotics, space, railways telecommunications, ...; foundational scheduling and predictability questions, such as schedulability analysis, synchronization protocols, multi-core scheduling, ...; static and dynamic techniques for resource demand estimation, such as classic worst-case execution time (WCET) analysis, ...; formal methods for the verification and validation of real-time systems, such as model checking, computer-assisted proofs, ...; the interplay of timing predictability and other non-functional qualities, such as reliability, quality of control, testability, scalability, ...; etc. Deadline for submissions: May 12, 2025 (industrial challenge), May 15, 2025 (workshop contributions), June 24, 2025 (real-time pitches). Deadline for early registration: June 13, 2025.

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. Deadline for submissions: April 1, 2025 (regular and short papers, workshop papers, fast abstracts, industry track, posters).

August 06-07 — 1st **Reliability and Maintainability Symposium European Edition** (RAMS Europe'2025), Amsterdam, the Netherlands. Deadline for registration: May 26, 2025 (authors), 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: April 5, 2025 (abstracts, full papers), June 19, 2025 (posters, demos), July 7, 2025 (tutorials). Deadline for early registration: July 18, 2025.

August 25-30 — 36th **International Conference on Concurrency Theory** (CONCUR'2025), Aarhus, Denmark. Co-located with QEST+FORMATS and FMICS as part of CONFEST 2025 Topics include: theory and practice of concurrent systems; verification and analysis techniques for concurrent systems (abstract interpretation, model checking, race detection, run-time verification, static analysis, testing, theorem proving, type systems, security analysis, ...); distributed/parallel algorithms and concurrent data structures (design, analysis, complexity, correctness, fault tolerance, reliability, availability, consistency, ...); theoretical foundations, tools, and empirical evaluations of architectures, execution environments, and software development for concurrent systems such as multiprocessor and multi-core architectures; compilers and tools for concurrent programming; programming models such as component-based, object-oriented, ...; etc. Deadline for submissions: April 3, 2025 (abstracts), April 9, 2025 (papers).

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. Deadline for submissions: April 4, 2025 (abstracts), April 11, 2025 (papers), April 18, 2025 (artifacts).

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. Deadline for submissions: June 10, 2025 (papers).

September 09-12 — 44th **International Conference on Computer Safety, Reliability and Security** (SafeComp'2025), Stockholm, Sweden. Topics include: all aspects related to the development, assessment, operation, and maintenance of safety-related and safety-critical computer systems; fault detection and recovery mechanisms; safety guidelines and standards; safety/security co-engineering and trade-offs; safety and security qualification, quantification, assurance and certification; threats and vulnerability analysis; model-based analysis, design, and assessment; formal methods for verification, validation, and fault

tolerance; testing, verification, and validation methodologies and tools; etc. Domains of application include: railways, automotive, space, avionics & process industries; highly automated and autonomous systems; telecommunication and networks; safety-related applications of smart systems and IoT; critical infrastructures, smart grids, SCADA; medical devices and healthcare; surveillance, defense, emergency & rescue; logistics, industrial automation, off-shore technology; education & training. Deadline for submissions: May 5, 2025 (workshop papers).

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

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: April 18, 2025 (special sessions), May 21, 2025 (papers), 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. Deadline for submissions: May 30, 2025 (papers).

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. Deadline for submissions: June 10, 2025.

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: June 1, 2025 (late breaking track papers).

Sep 30 – Oct 1 — 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. Deadline for early registration: June, 2025.

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 12-18 — ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2025), Singapore. Deadline for submissions: May 8, 2025 (tutorials).

☺ Oct 12-25 — **Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2025).

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: May 5, 2025 (abstracts), May 12, 2025 (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: May 31, 2025 (papers).

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: June 23, 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: May 29, 2025 (2nd round), 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. Deadline for submissions: June 6, 2025 (abstracts), June 20, 2025 (papers).

☺ 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. Deadline for submissions: June 16, 2025 (technical track talks).

☺ 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: May 30, 2025 (abstracts), June 6, 2025 (papers), 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: June 6, 2025 (abstracts), June 13, 2025 (papers), September 26, 2025 (posters).

☺ 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: May 22, 2025 (papers).

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

2026

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).

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: June 3, 2025 (round 1), October 16, 2025 (round 2).

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: June 23, 2025 (workshops), 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).



Advance Information

The 29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025) returns to Paris, France, from the 10th to the 13th of June. The conference 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 Ada and Reliable Software in general, into academic education and research, and industrial practice.

The conference is an established international forum for providers, practitioners and researchers in reliable software technologies. The conference presentations will illustrate current work in the theory and practice of developing, running and maintaining challenging long-lived, high-quality software systems for a variety of application domains including manufacturing, robotics, avionics, space, transportation. The conference schedule comprises two keynotes and an invited talk, a journal track, an industrial track, a work-in-progress track, parallel tutorials, and satellite workshops. Participants include practitioners and researchers from industry, academia and government organizations active in the promotion and development of reliable software. The conference program includes two core days with special sessions featuring presentations of invited experts, peer-reviewed academic papers, industrial presentations, and work-in-progress talks.

Invited Speakers

- Keynote Talk on "*Building a Safer Open Source Supply Chain with Software Heritage, the Great Library of Source Code",* by **Stefano Zacchiroli,** Télécom Paris, Polytechnic Institute of Paris, France
- Keynote Talk on "*Certification of Machine Learning Based System - Contribution of Formal Verification",* by **Claire Pagetti,** ONERA, France
- Invited Talk on "*Language Ownership: Key Moments in the Lifetime of Ada",* by **Tullio Vardanega,** University of Padua and Ada User Society, Italy

Technical Presentations

The conference technical days will provide six sessions of presentations related to the conference tracks, including presentations of the 6 research articles which have passed the first round of reviews in the journal track, 3 presentations of industrially relevant research and development, and 9 work-in-progress presentations. Please see the conference website for information.

Tutorials

The following tutorials will take place on Tuesday, June 10th:

- "Developing Bare-metal Embedded Software in SPARK Ada for 64-bit ARM Platforms", **J. Germán Rivera**, Tesla, USA
- "Multiple Double and Multiword Arithmetic", **Jan Verschelde**, University of Illinois at Chicago, USA
- "Unleash the power of Ada Generics", **Gautier de Montmollin**, Ada-Switzerland, Switzerland
- "Turning your Programming Language into a Modeling Language", **S. Tucker Taft**, AdaCore, USA
- "Introduction to the ALiRe Package Manager", **Quentin Dauprat**, Université de Caen, Normandie, France

Co-Located Workshops

On Friday, June 13th there will be 3 workshops:

- 10th DeCPS workshop on “Challenges and new Approaches for Dependable and Cyber-Physical Systems Engineering”
- 4th ADEPT: AADL by its practitioners
- Ada Developers Workshop

Social Events

The program includes long coffee breaks, providing the opportunity for participants to discuss their work, and network. Lunches will be served at the conference location, from Tuesday to Friday, providing further interaction opportunities. Furthermore, there will be a Welcome Reception event, a Conference Banquet, and a post-conference chill-out event.



The usual welcome reception is scheduled for Tuesday, June 10, from 19:00 to 21:00. It will take place at the Table du Luxembourg, a nice restaurant right in the middle of the Luxembourg garden, a Parisians' favorite! While going to the restaurant, don't miss the Medici fountain, then pass in front of Luxembourg Palace, watch the children playing with little boats in the fountain, and admire the famous statues of the queens of France!





The conference banquet is scheduled for Wednesday, June 11th at 20:00. It will take place 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. We will have dinner in the beautiful dining room on the first floor, an authentic turn of the 20th century decoration.

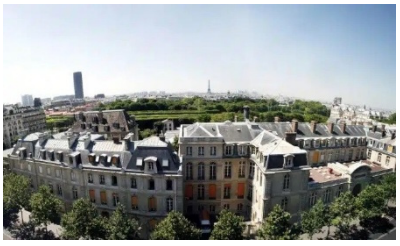


The organization also prepared a chill out event as a culmination party for this year's main conference. After all the exciting presentations at the conference, have a little chat with colleagues in a casual typical café: the Café Latin. This is an informal dinner with a separate registration. For those who want to get a feeling of the Latin Quarter, the conference website provides an itinerary that will show you the main milestones of the neighbourhood while walking to the Café Latin.



Conference Venue

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



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



Join Ada-Europe!

Become a member of Ada-Europe and **support Ada-related activities** and the future **development of the Ada programming language**.

Membership benefits include **receiving the quarterly Ada User Journal** and a substantial **discount when registering for the annual Ada-Europe conference**.

To apply for membership, visit our web page at



http://www.ada-europe.org/**join**

ADEPT 2024 Workshop Summary

Hai Nam Tran, Frank Singhoff
University of Brest, Lab-STICC UMR CNRS 6285, Brest, France; email: firstname.lastname@univ-brest.fr

Jérôme Hugues
Software Engineering Institute, Carnegie Mellon University, USA; email: jhugues@andrew.cmu.edu

Emmanuel Grolleau
LIAS, ISAE-ENSMA, France; email: grolleau@ensma.fr

Leandro Buss Becker
Graduate Program on Automation and Systems Engineering, Federal University of Santa Catarina (UFSC), Florianópolis, Brazil; email: leandro.becker@ufsc.br

Leonidas Kosmidis
Barcelona Supercomputing Center (BSC) and Universitat Politecnica de Catalunya (UPC); email: leonidas.kosmidis@bsc.es

Monica Michelle Villegas Arias
Federico Santa María Technical University, Chile; email: monica.villegas@sansano.usm.cl

Bruce Lewis, Steve Vestal
Galois, Inc., USA; email: firstname.lastname@galois.com

Pierre Dissaux
Ellidiss Technologies, 24 quai de la douane, 29200 Brest, France; email: pierre.dissaux@ellidiss.com

Abstract

The Architecture Analysis and Design Language (AADL) is a SAE standard for modeling both hardware and software architecture of embedded systems. Widely embraced by stakeholders in critical real-time embedded systems, the AADL standard is used to address a large set of concerns including performance (latency, schedulability), safety, and security. The ADEPT workshop aims to present and report on current projects in the field of design, implementation, and verification of critical real-time embedded systems where AADL is a first-citizen technology. This article is a summary of the third edition of the workshop in 2024.

Keywords: AADL, critical embedded real-time systems, design, implementation and verification

1 Introduction

The Architecture Analysis and Design Language (AADL) is a SAE standard for modeling both hardware and software architecture of embedded systems. [1]. The AADL standard is now a mature standard for modeling critical real-time embedded systems. It is employed by numerous stakeholders in the domain to address a large set of concerns: safety [2], security [3], or performance (latency, schedulability) [4] but also code generation [5, 6]. One key strength of AADL as a language is the set of tools that provide those analysis capabilities.

The ADEPT workshop aims to present and report on current projects in the field of design, implementation and verification of critical real-time embedded systems where AADL is a first citizen technology. It is also an opportunity for AADL beginners to meet experienced AADL practitioners.

In 2024, the workshop was a full day workshop. It was co-located with the 28th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2024) at Barcelona, Spain. The workshop gathered 15 participants, with 9 presentations.

2 Workshop Program

The workshop program includes an introduction about the AADL standard, its ecosystem and the ongoing standardization activities. The technical program is organized in 3 sessions: (1) Unmanned Aerial Vehicles, (2) Model-based System Engineering, and (3) AADL tools update.

2.1 Unmanned Aerial Vehicles

This first session focuses on the use of AADL in the development process of Unmanned Aerial Vehicles (UAV). In [7], the author discusses the main points where AADL lacked to provide the required concepts for drone manufacturers to assess the safety of their autopilots with custom components. The return of experience is in the scope of the Comp4Drones ECSEL JU European project which aims to provide a framework of key enabling technologies for safe and autonomous drone applications.

In [8], the authors describe the integration of AADL in a formal verification process to create a safe continuous control architecture for a UAV. The software architecture is first modeled in AADL then transformed into a set of timed automata in order to perform formal verification with UPPAAL.

2.2 Model-based System Engineering

The second session of the workshop focuses on the integration of AADL in a MBSE process. In [9], the authors present how modeling technologies like TASTE and AADL are employed within the METASAT toolchain to enable efficient development of satellite software components. The METASAT project addresses these challenges through an innovative approach combining model-based engineering (MBE) with open architecture hardware for satellite system development.

In [10], the authors present the AQDT-GEN framework that automatically builds the digital twin of an IoT system using the configurations, parameters and specifications of its AADL model. AQDT-GEN transforms architectural specifications into required deployment files, parameters, and launching scripts, for both emulation and simulation, using the AADL model and the generation rules.

In [11], the authors present the CosiCosi simulator whose the aim is to characterize the temporal behavior of complex systems modeled in AADL. The simulator allows us to simulate the temporal behavior of the AADL model components, respecting the AADL runtime specification as defined in the standard.

2.3 AADL Tools Update

In the third session, workshop participants present, discuss and provide updates on several AADL tools and toolsets including INDIGO [12], CAMET [13], Ellidiss's AADL tools [14], and Cheddar [15].

- INDIGO (INsight to Diverse Information using Graphs and Ontologies) is an open collaborative model-based engineering ecosystem that integrates diverse modeling languages and tools, including SysML-V2 and AADL, through graph-based and ontology-driven approaches.
- CAMET (Curated Access to Model-based Engineering Tools) is a library of MBSE tools—primarily centered on AADL and the OSATE environment
- Ellidiss's AADL tools include Stood for AADL, AADL Inspector, and an AADL Visual Studio Code extension. The presentation also highlights ongoing work to aligning AADL with SysML v2 via a dedicated library that preserves AADL semantics
- Cheddar is a real-time scheduling analyzer. In [15], the authors provide guidelines and explain schedulability analysis methods implemented in Cheddar to analyze AADL models of multiprocessor systems.

3 Conclusion

AADL is a set of SAE international standards that aims to improve the quality of critical real-time embedded systems design. The objective of the ADEPT workshop was to encourage discussion between members of the AADL community and to provide location to share experiences on AADL and its ecosystem.

Acknowledgments

We would like to thank Ellidiss Technologies for supporting the registration fees for the speakers of the workshop.

References

[1] S. I. (Society), *Architecture Analysis & Design Language (AADL) AS5506C*. SAE International, 2017.

[2] J. Delange and P. Feiler, "Architecture fault modeling with the AADL error-model annex," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 361–368, IEEE, 2014.

[3] R. Ellison, A. Householder, J. Hudak, R. Kazman, and C. Woody, "Extending AADL for security design assurance of cyber-physical systems," *CMU/SEI Report*, 2015.

[4] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Scheduling and memory requirements analysis with aadl," *ACM SIGAda Ada Letters*, vol. 25, no. 4, pp. 1–10, 2005.

[5] J. Hugues, B. Zalila, L. Pautet, and F. Kordon, "From the prototype to the final embedded system using the ocarina AADL tool suite," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 4, pp. 1–25, 2008.

[6] F. Cadoret, E. Borde, S. Gardoll, and L. Pautet, "Design patterns for rule-based refinement of safety critical embedded systems models," in *2012 IEEE 17th International Conference on Engineering of Complex Computer Systems*, pp. 67–76, IEEE, 2012.

[7] E. Grolleau, "UAV autopilot architectures versus AADL." Presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

[8] L. B. Becker, F. S. Goncalves, E. F. Broering, H. A. Misson, and L. Cordeiro, "Safe UAV Continuous-Control Architecture Design," *Ada User Journal*, vol. 46, no. 1, pp. 50–53, 2025. Also presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

[9] A. Calderón, I. Yarza, S. Sinisi, L. Lazzara, V. Di Valerio, G. Stazi, L. Kosmidis, M. Trompouki, A. Ulisse, A. Amonarriz, and P. Onaindia, "TASTE and AADL in the METASAT Model-Based Engineering Workflow," *Ada User Journal*, vol. 46, no. 1, pp. 54–57, 2025. Also presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

[10] M. M. Villegas Arias, H. Astudillo, and F. Giraldo, "AQDT-GEN: AADL and QEMU-based digital twin generation for IoT testing." Presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

[11] S. Rubini, S. Levieux, E. Cariou, F. Singhoff, H. N. Tran, and G. Le Pluart, "Executable AADL Models for Early System Qualification Test," *Ada User Journal*, vol. 46, no. 1, pp. 58–62, 2025. Also presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

[12] S. Vestal and E. Sandberg, “INDIGO - collaborative MBE across diverse system models including AADL.” Presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

[13] B. Lewis and R. Whillock, “CAMET AADL tools.” Presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

[14] P. Dissaux, “Ellidiss tools for AADL modeling and verification, and current state of the SysML v2 library for AADL support.” Presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

[15] F. Singhoff, S. Rubini, H. N. Tran, and S. Levieux, “AADL Modeling and Schedulability Analysis of Multiprocessor Architectures,” *Ada User Journal*, vol. 46, no. 1, pp. 63–71, 2025. Also presented in the ADEPT 2024 workshop, Barcelona, Spain, 2024.

Safe UAV Continuous-Control Architecture Design

L. B. Becker, E. F. Broering

Graduate Program on Automation and Systems Engineering, Federal University of Santa Catarina (UFSC), Florianópolis, Brazil; email: leandro.becker@ufsc.br, eltonbroering@gmail.com

F. S. Gonçalves

Extension, Research and Education Department, Federal Institute of Santa Catarina (IFSC) - Campus Tubarão, Brazil; email: fernando.goncalves@ifsc.edu.br

H. A. Misson

Polytechnique Montréal, Canada; email: henriquemisson@gmail.com

L. Cordeiro

Department of Computer Science, University of Manchester, UK; email: lucas.cordeiro@manchester.ac.uk

Abstract

Unmanned Aerial Vehicles (UAVs) are often classified as safety-critical systems due to the severe implications a failure might have. Ensuring safety guarantees is therefore paramount in UAV design. This paper introduces a solution to provide safety guarantees during the design of the continuous-control architecture, a critical component. Formal verification (FV) techniques are employed to detect software errors and verify the architecture's ability to meet real-time system requirements. The first verification round utilizes the ESBMC model checker, focusing on individual elements of the architecture, particularly the continuous-control algorithm. The second verification process employs the UPPAAL model checker to conduct a schedulability analysis of the proposed architecture. Finally, a runtime monitoring analysis is performed using the newly developed RMLib tool. This approach was successfully applied in the design process of a UAV, identifying and correcting various design and implementation issues, as detailed in the paper.

Keywords: UAV, Formal Verification, Model Checking, Schedulability analysis, Time-correctness, Safety, Implementation-errors analysis, Runtime Monitoring.

1 Introduction

Failures in UAV applications pose significant risks to human lives, natural resources, and expensive equipment. These applications are subject to stringent real-time constraints, meaning deadlines cannot be missed without causing erroneous actions. Therefore, it is crucial to enhance the design process with methods that ensure the system's time-correctness. Formal verification becomes essential in the UAV design process, especially during the early development phases [1].

This paper summarizes the work in [2], which reports on the application of formal verification to create a safe continuous-control architecture for a UAV. The goal is to ensure the safety and time-correctness of the software components used. This initiative is not intended to be the sole method for UAV design using formal verification but rather a novel approach that has proven valuable in the UAV design process.

The core elements of our verification process include the Efficient SMT-based Bounded Model Checker (ESBMC) [3, 4], which detects user-specified assertion failures and other issues such as out-of-bounds array access, illegal pointer dereferences, integer overflows, NaN (floating-point), division by zero, and memory leaks. The UPPAAL tool [5] is used to formally verify the scheduling algorithm designed for the continuous-control architecture. The RMLib tool [6] supports runtime monitoring (RM) on top of FreeRTOS, allowing for both offline and online analysis of execution traces.

The reminder parts of the paper are organized as follows. Section 2 presents the proposed FV process, describing each verification step. In section 3 we detail the RM tool. The application of the FV process and the RM tool for the design of the continuous-control architecture of a UAV is discussed in Section 4. Finally, we present our conclusions and possible directions for future work perspectives in Section 5.

2 Proposed Formal Verification Process

The proposed FV process begins applying Model-Checking (MC) by means of two different static verification techniques/-tools. Firstly, static code verification in the the continuous-control algorithm is performed, searching for implementation errors. Next, we conduct schedulability analysis using as input an AADL model. In what follows, these two techniques are presented in detail.

2.1 Verifying Errors in Control Algorithms

This phase conducts model checking on the continuous-control algorithm's source code, utilizing the C programming language as input. The objective is to identify implementation errors, including but not limited to pointer safety, arithmetic overflow, and division-by-zero—issues that are often overlooked by conventional software testing methods [7].

The Efficient SMT-based Bounded Model Checker (ESBMC) [3] is employed for this verification process. ESBMC is an open-source, permissively licensed tool grounded in satisfiability modulo theories (SMT), designed to verify single and multi-threaded C/C++ programs. Within ESBMC, diverse software verification techniques and solvers, such as incremental BMC, *k*-induction, and SMT solvers like Bitwuzla, Boolector, CVC4, MathSAT, Yices, and Z3, can be applied for program analysis. ESBMC is versatile in detecting various software errors, including out-of-bounds array access, illegal pointer dereferences, integer overflows, NaN (Floating-point), division-by-zero, and memory leaks [8].

2.2 Schedulability analysis

This phase performs model checking using as input the software architecture model (AADL model) developed for the target system. However, it requires the AADL model to be transformed into a set of Timed Automata (TA). Therefore, we adopt the model transformation mechanism described in [9], which uses UPPAAL [5] to perform the formal verification.

UPPAAL allows the modeling of systems in TA extended with integer variables, structured data types, and channel synchronization. This consists of finite state machines with a clock, in which the model is composed of states (locations) that represent situations or modes of operation and of transitions that connect two or more states. This network of states and transitions forms the system's behavior, where for each transition, it is possible to define a set of rules and conditions that the system must meet to enable a state change.

Properties to be verified in UPPAAL are formalized using the Timed Computation Tree Logic (TCTL) language [10], which is a fragment of CTL [11] that uses clock variables and clock constraints to specify timing behaviors reasoning about properties of the TA. TCTL allows constructing the queries using the quantifiers *A* (for all paths) and *E* (exists a path), the temporal operators $\diamond$ (eventually) and $\Box$ (always), beyond the Boolean connectives (*and, or, implication*). Properties are typically organized in patterns, as follows:

- **Reachability**: it allows us to evaluate if there is a path where at least one state satisfies a given α property: $E \diamond \alpha$
- **Safety**: this property states that "something bad never happens", meaning that the system should never reach the state where this property is satisfied: $A \Box \alpha$
- **Liveness**: this enables checking whether, considering all paths, states exist that satisfy a given α property: $A \diamond \alpha$

To avoid state explosion situations, some design techniques can be applied. One of these techniques is creating an Observer Automata, restricting a system's legitimate behavior to those accepted by such observers. This technique is used in the developed schedulability analysis model.

3 Runtime Monitoring Library

RMLib [6] is a lightweight and non-intrusive solution for implementing runtime monitoring (RM) in embedded systems, allowing users to perform customization based on specific needs. Although architecture-independent and compatible with multiple platforms, the proposed tool was developed to support the FreeRTOS operating system. Overall, the developed library allows addressing the following issues:

1. Monitor if all tasks in the system are executed without missing deadlines. This is important to assess the real-time performance of the system and validate the scheduling and timing properties.
2. Measure (and reason about) the execution times of the tasks. This helps identify potential bottlenecks or performance issues and provides insights into the system's timing behavior.

RMLib can support two distinct operation modes: online verification and offline verification. In offline verification mode, the monitors run concurrently with the application code, collecting timestamps at specific points of interest. These timestamps are then stored for later exportation and evaluation of monitored constraints using external software. Offline verification is less intrusive and imposes less overhead on the system, but it is not helpful to identify situations where immediate response is required.

On the other hand, online verification mode operates similarly to offline verification regarding event monitoring. However, verifying system restrictions occurs online and does not require transferring all monitored events – only the verification results are exported. Online verification offers additional features such as timestamp analysis, exporting only cases of timing violations, and exporting system information when violations occur. However, it imposes the system's overhead and, depending on the configuration, it can impact the system's performance.

4 Case Study: UAV Continuous-control Architecture Design

This section summarizes the application of the proposed FV process within the context of a project aiming for designing a UAV, whose dynamics are detailed in [12]. More specifically, it aims to present the design of a safe continuous-control architecture suitable not just for the UAV under design in our project but also for other CPS, as further detailed.

4.1 Verification in ESBMC

The *Code Verification Phase* targeted the Linear Quadratic Regulator (LQR) control algorithm described in [12], developed in the C programming language and previously validated using the Simulink tool. The program analysis is conducted in an open-loop fashion, with a single execution of the control loop for each input configuration. ESBMC tests all possible values of the input variables within the control loop. These variables consist of 48 floats, 18 integers, and 4 unsigned integers. These values originate directly from UAV sensors or indirectly from such measurements. To enhance the efficiency of the verification process, bounds (upper and lower limits) were defined for each of these 70 variables based on the simulation analysis conducted in the ProVant project.

ESBMC has found violations related to arithmetic overflow (AO) and "Not a Number" (NaN). In particular, AO occurs when the result of an arithmetic operation exceeds the range

that can be represented by the data type used to store the result. In this case, ESBMC has found a computation result that is too large to be represented within the constraints of the underlying control algorithm data type, causing the value to wrap around or become undefined. For the "Not a Number" violation, which is a special floating-point value that represents the result of an undefined or unrepresentable operation, ESBMC has found a mathematical operation in the underlying control algorithm that cannot produce a meaningful result.

Overall, we can also analyse counterexamples generated by ESBMC against the new instrumented program to pinpoint faulty lines in the UAV controller implementation. This involves searching for diagnostic values corresponding to actual lines in the embedded UAV software. Such an approach can further enhance debugging processes for UAV-embedded software by indicating which lines require correction and identifying the values leading to successful execution. However, we defer the exploration of this research direction to future work focused on automated UAV software fault localization and debugging.

4.2 Verification in UPPAAL

The initial AADL model was transformed into a set of Timed Automata (TA) to be processed by UPPAAL. As a result of this transformation, templates that describe our architecture were generated. It consists of an auxiliary scheduler TA and five instances of the TA representing the generic thread behavior, with specific parameters for each thread.

To avoid race conditions among shared data, thread preemptions are not allowed in this model, thus creating implicit mutual exclusion areas. Such a solution mixes preemptable and non-preemptable threads, typically making analytical schedulability analysis much more complex, sometimes impossible. However, this is not an issue in the present work, given the use of Model Checking. The AADL model was transformed into TA using the AADL–to–TA transformation tool presented in [9].

The set of properties to be formally verified must be specified in TCTL. The first property analyzes the schedulability of the task set, proving that there are no deadline misses, i.e., no thread reaches the *Error* state. The second property proves that the adopted strategy to avoid race conditions is successful, i.e., those threads that share memory (T1–T4) will not be preempted once they start executing. The third property relates to the *T1–Actuation* thread, which cannot have jitter to start executing. The derived property states that while in the Processing state, the thread general clock t is synchronized with the clock related to the execution time ax. This property does not hold for the other threads (T2–T5). The last verified property verifies the order in which the threads are executed so that it becomes possible to prove that they are executed exactly as planned. In other words, it is necessary to guarantee that higher-priority threads will always have preference over lower-priority ones. To verify this property, it was necessary to create an observer, which consists of an additional mechanism to "count" the thread's execution. It is indeed an overhead for the verification process, increasing the number of states. However, it does not significantly increase the execution time of the present study.

The conducted schedulability analysis, which shows that the proposed task set is indeed schedulable, would be very difficult to be done in analytical means. This comes from the fact that the adopted parameters for threads do not suit a single scheduling policy. Taking for exemple the well-known Cheddar [13] scheduling-analysis tool, it does not allow to mix preemptable and non-preemptable threads. Instead, either one uses a scheduling policy that does not allow preemptions, or preemptions should be allowed for all tasks.

For instance, the designer can select a preemptable scheduling policy and establish critical sections that last the execution of those non-preemptable threads. However, there is no means to make an analytical schedulability analysis for such a situation, so only simulations would be possible. However, simulations do not provide strong guarantees. Besides, the results (simulation logs) can be challenging to be analyzed.

4.3 Runtime Monitoring Results

Our RTMLib tool was used to monitor the timing behavior from all 5 threads. The conducted analysis involved examining the tasks' executions and judging if they were executed within their WCET and deadline restrictions. The verdict is determined based on the following criteria:

- If the difference (Δ) between task's finish and release times is less than or equal to the task's deadline, the verdict is $true$. Otherwise, if Δ exceeds the deadline, the verdict is $false$.
- Another Δ is calculated as the difference between tasks' start and finish times. If it is less than or equal to the WCET the verdict is $true$, otherwise it is $false$.

Another specific point of analysis relates to the Actuation task ($T1$), given the need for periodic reception of a command by the ESC. Task $T1$ is the most critical in our proposed architecture because the UAV might fall if it does not work properly (within the defined time-limits).

In summary, the results from all experiments (on the different operation modes of our tool) showed no timing-constraints violations during the executions. It can also be observed that, in most cases, the Actuation task ($T1$) is executed immediately after it is released. The maximum waiting time observed was of 3 ms, which indeed is not a problem, as it allows $T1$ to complete its execution much ahead of its deadline.

4.4 Reusing the Proposed Architecture

The proposed continuous-control architecture can be reused in different CPS applications. For instance, we have a new UAV is under construction, which will benefit from using this proposal. The first essential point to reusing the proposed architecture is designing a proper control algorithm. This implies modifying the inputs and outputs of the control algorithm. Possibly, the algorithm itself will also change. Consequently, all the steps related to the control algorithm analysis using ESBMC must be re-conducted.

Modifying the inputs and outputs of the control algorithm necessarily affects the connections among the AADL process. However, the most critical adjustment for performing the

schedulability analysis relates to properly modifying the timing parameters of the system, especially the execution times.

When reusing the proposed architecture in the same CPS domain, such as using it in a new UAV, it is likely that the periods and deadlines of the core threads (T1,T2,T3) will not change, given that the system dynamics are mostly the same. However, when the system dynamics are different, like the case in the smart energy grid, it is required that such core threads' periods and deadlines change as a multiple of the original ones. It is expected that such new periods will be higher, not smaller, than the originals. This implies increasing the critical instant at a multiple of the $12ms$ one used in the study previously presented. Once these adjustments are finished, the designer can conduct the schedulability analysis with UPPAAL.

5 Conclusions and Future Works

This paper presents a formal verification (FV) process aimed at enhancing the safety of software architecture within Cyber-Physical Systems. The FV process is utilized to develop a safe continuous-control architecture for a UAV. The proposed architecture is sufficiently generic to be applicable to other continuous-control/CPS applications.

The FV process has proven effective in improving the safety of the continuous-control architecture. ESBMC has been instrumental in identifying implementation errors in the continuous-control algorithms. Beyond the benefits achieved in this specific project, the consistent use of ESBMC by the control algorithm design team can enhance programming practices, enabling earlier detection of errors. Additionally, the schedulability analysis infrastructure developed using UPPAAL provides safety guarantees for the designed architecture. The model-checking-based schedulability analysis surpasses the limitations of existing analytical techniques by allowing full processor utilization (up to 100%) while accommodating both preemptable and non-preemptable tasks and providing counter-examples when the task set is not schedulable.

The developed Runtime Monitoring library successfully collected and analyzed events related to tasks' timing properties, such as periodicity, execution time, and deadline misses. Future work should focus on making the library more generic to handle various event types.

Further efforts should be directed in two areas. First, enhancing ESBMC to facilitate the identification of faulty lines in the analyzed code, as this task is currently labor-intensive. Second, providing runtime verification support to integrate formal verification into the proposed process comprehensively, thereby achieving maximum safety levels. Initial efforts in this direction have already begun, targeting the FreeRTOS operating system [14].

References

[1] R. Austin, *Unmanned aircraft systems: UAVS design, development and deployment*. New York, U.S.: John Wiley & Sons, 2011.

[2] L. B. Becker, F. S. Gonçalves, E. F. Broering, H. A. Misson, and L. Cordeiro, "Applying formal methods to build a safe continuous-control architecture for an unmanned aerial vehicle," *PREPRINT available at Research Square [https://doi.org/10.21203/rs.3.rs-3668418/v1]*, 2023.

[3] M. R. Gadelha, F. R. Monteiro, J. Morse, L. C. Cordeiro, B. Fischer, and D. A. Nicole, "ESBMC 5.0: An industrial-strength C model checker," in 33^{rd} *ACM/IEEE Int. Conf. on Automated Software Engineering (ASE'18)*, (New York, NY, USA), pp. 888–891, ACM, 2018.

[4] M. R. Gadelha, F. Monteiro, L. Cordeiro, and D. Nicole, "Esbmc v6. 0: Verifying c programs using k-induction and invariant inference: (competition contribution)," in *Tools and Algorithms for the Construction and Analysis of Systems: 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part III 25*, pp. 209–213, Springer, 2019.

[5] G. Behrmann, A. David, and K. G. Larsen, *Sfm-rt 2004*, ch. A Tutorial on Uppaal, pp. 200–236. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[6] E. F. Broering and L. B. Becker, "Applying runtime verification in real-time systems with freertos," in *2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pp. 1–6, 2022.

[7] K. M. Alshmrany, M. Aldughaim, A. Bhayat, and L. C. Cordeiro, "Fusebmc v4: Smart seed generation for hybrid fuzzing - (competition contribution)," in *25th International Conference on Fundamental Approaches to Software Engineering - FASE 2022* (E. B. Johnsen and M. Wimmer, eds.), vol. 13241 of *LNCS*, pp. 336–340, 2022.

[8] F. R. Monteiro, M. Garcia, L. C. Cordeiro, and E. B. de Lima Filho, "Bounded model checking of C++ programs based on the qt cross-platform framework," *Softw. Test. Verification Reliab.*, vol. 27, no. 3, 2017.

[9] F. S. Gonçalves, D. Pereira, E. Tovar, and L. B. Becker, "Formal verification of aadl models using uppaal," in *2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pp. 117–124, IEEE, 2017.

[10] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[11] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Workshop on Logic of Programs*, pp. 52–71, Springer, 1981.

[12] R. Donadel, G. Raffo, and L. Becker, "Modeling and control of a tiltrotor uav for path tracking," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3839–3844, 2014. 19th IFAC World Congress.

[13] F. Singhoff, J. Legrand, and L. N. L. Marcé, "Cheddar : a flexible real time scheduling framework," *ACM SIGAda Ada Letters*, vol. 24, no. 4, pp. 1–8, 2004.

[14] FreeRTOS.org, "Freertos: Real-time operating system for microcontrollers," May 2022.

TASTE and AADL in the METASAT Model-Based Engineering Workflow

Alejandro J. Calderón, Irune Yarza, Aitor Amonarriz, Peio Onaindia
Ikerlan Technology Research Centre; email: {ajcalderon, iyarza, aamonarriz, ponaindia}@ikerlan.es

Stefano Sinisi, Lorenzo Lazzara, Valerio Di Valerio, Giulia Stazi, Alessandro Ulisse
Collins Aerospace; email: {stefano.sinisi, lorenzo.lazzara, valerio.divalerio, giulia.stazi, alessandro.ulisse}@collins.com

Leonidas Kosmidis, Matina M. Trompouki
Barcelona Supercomputing Center (BSC); email: {leonidas.kosmidis, matina.trompouki}@bsc.es

Abstract

The increasing complexity of satellite systems, in conjunction with the need for compliance with ECSS standards, demands innovative design methodologies. The METASAT project introduces a model-based engineering workflow that integrates open-architecture hardware and advanced software virtualisation layers. This paper presents how modelling technologies like TASTE and AADL are employed within the METASAT toolchain to enable efficient development of satellite software components. We detail the integration of these tools within the METASAT workflow, highlighting their role in system architecture design, validation, and code generation. The approach demonstrates how model-based engineering techniques can significantly reduce development costs and timelines while improving system performance and dependability for future satellite missions.

1 Introduction

The evolution of satellite technology has led to a phase of significant complexity in spacecraft design and operation. Modern satellites increasingly incorporate sophisticated capabilities driven by artificial intelligence, real-time analytics, and advanced sensing technologies. This growing complexity presents significant challenges for engineers tasked with developing reliable, standards-compliant satellite systems within reasonable time frames and budgets.

The satellite industry faces a particular challenge in on-board software engineering, where emerging technologies require robust foundations to function effectively. Three critical requirements emerge: hardware platforms with sufficient computational capacity, software layers that efficiently utilise resources while maintaining dependability, and design methodologies adapted to the unique challenges of space systems development.

Launched in January 2023, the METASAT Horizon Europe project addresses these challenges through an innovative approach combining model-based engineering (MBE) with open architecture hardware. The project emphasizes that software virtualisation layers, specifically hypervisors, can enable standards compliance even on high-performance computing platforms. A key element of this vision is a dedicated toolchain for these advanced hardware/software configurations.

Without such innovations, the satellite industry risks facing prohibitive development costs and extended timelines that could compromise competitiveness and potentially compromise system reliability. The METASAT approach demonstrates how well-designed model-based techniques can transform the development process, making it more efficient while enhancing system dependability. A more detailed description of the METASAT project can be found in [1].

This paper examines the integration of two key technologies within the METASAT framework: TASTE (The ASSERT Set of Tools for Engineering) and AADL (Architecture Analysis and Design Language). We explore how these technologies function within the METASAT toolchain and workflow, emphasizing their contributions to system architecture design, validation processes, and code generation. Our review begins with an overview of the METASAT project itself, followed by detailed discussions of TASTE and AADL implementation, and concludes with reflections on the implications for future satellite engineering practices.

2 Overview of the METASAT Project

The METASAT initiative represents an innovative approach to satellite system development, integrating cutting-edge hardware capabilities with sophisticated software frameworks through a comprehensive model-based engineering methodology. This section presents the innovative hardware-software architecture of the project and its MBE toolchain.

2.1 Hardware and Software Architecture

At the core of the METASAT hardware platform lies a multicore implementation of the NOEL-V space processor, built on the open RISC-V architecture. This architecture is strategically enhanced with the SPARROW AI SIMD accelerator [2], a specialised component designed to handle the computational demands of advanced artificial intelligence algorithms

in space environments. Complementing this processing capability, the platform incorporates a Vortex GPU (Graphics Processing Unit) [3] connected to the central processor cluster via a high-throughput AXI interface.

The software architecture integrates the XtratuM hypervisor as its core element, providing a virtualisation layer that creates strictly isolated partitions for applications with varying criticality levels, which is a crucial requirement for space systems. Within this framework, partitions can operate either under the RTEMS real-time operating system or as baremetal implementations using the XtratuM Runtime Environment (XRE). This architecture enables multiple partitions to share specialised hardware resources including the AI accelerator and GPU while maintaining the isolation properties essential for safety-critical systems.

2.2 The Model-Based Engineering Toolchain

The METASAT toolchain integrates several specialised technologies to support a comprehensive development workflow from initial requirements through to deployment. The framework is built around TASTE [4] [5], complemented by some MathWorks tools.

TASTE provides an open-source environment specifically designed for embedded real-time systems development. Its particular strength lies in its ability to integrate diverse application modules, either hand-coded in languages like C and Ada, or automatically generated through external tools such as MATLAB/Simulink. The framework uses formal languages including AADL [6] for architectural description and ASN.1 [7] for precise data type specification, ensuring consistency and verifiability throughout the development process.

The MathWorks toolset contributes to the toolchain with essential capabilities for requirements management, model validation, and code generation. Key elements include the Requirements Toolbox for formalising and tracking system requirements, Simulink for its simulation capabilities, Simulink Test for automated validation, Embedded Coder for generating optimised C/C++ implementations, and GPU Coder for producing specialised code targeting hardware accelerators.

This integrated toolchain creates a unified environment supporting the entire development lifecycle, from initial concept through to verification on virtual or physical hardware platforms.

3 TASTE and AADL in the METASAT Toolchain

The integration of TASTE and AADL within the METASAT toolchain represents a practical approach to addressing the architectural challenges of modern satellite system development. This section explores how these technologies collaborate to provide a robust framework for model-based engineering of complex space systems.

3.1 TASTE: Orchestrating Heterogeneous System Development

TASTE serves as the central orchestration framework within the METASAT toolchain, providing a complete environment for developing embedded real-time systems across their entire lifecycle. Unlike conventional development environments, TASTE excels at bridging the gaps between diverse development approaches, a critical capability in the complex domain of satellite engineering.

One of the most valuable contributions of TASTE to the METASAT framework is its exceptional flexibility in component integration. The tool integrates application modules regardless of their origin, either manually created in traditional languages like C and Ada, or automatically generated from higher-level modelling environments such as MATLAB/Simulink. This versatility proves particularly valuable in satellite development, where different subsystems often emerge from specialised development processes with unique requirements and constraints.

The consistency of this integration is ensured through the implementation of formal languages by TASTE. AADL provides the architectural backbone, while ASN.1 delivers precise data type definitions with complete semantic information. This formal support enables rigorous verification of system properties throughout the development process.

For transforming these formal specifications into executable implementations, TASTE incorporates the Kazoo tool [8]. Kazoo synthesizes executable applications by combining functional blocks with automatically generated communication interfaces (glue code) according to the architectural specifications defined in AADL. This process relies on the Ocarina tool [9] for parsing and processing AADL models, creating a direct path from architectural description to implementation.

3.2 AADL: Formalising System Architecture

AADL occupies a fundamental position in the METASAT toolchain as the primary language for architectural specification. Designed specifically for real-time, safety-critical systems, AADL provides a standardised notation for describing both the structure and non-functional properties of complex systems—addressing precisely the challenges faced in satellite development.

Within the METASAT workflow, AADL serves dual purposes in capturing both functional and physical architectures. The functional architecture is constructed through the TASTE Interface View, which defines components and their interactions through provided and required interfaces. This creates a clear specification of system behaviour independent of implementation details.

The physical architecture takes shape through the TASTE Deployment View, which describes the target hardware platform and establishes the mapping between functional components and physical resources. This mapping is particularly critical in the METASAT context, where specialised hardware components like the NOEL-V processor, the SPARROW AI accelerator, and the Vortex GPU must be used efficiently while maintaining system integrity.

This separation between functional and physical concerns represents one of the most significant contributions of AADL to the METASAT approach. By maintaining distinct but formally related models of what the system does versus how it is implemented, AADL enables engineers to reason about these aspects independently while ensuring their compatibility through formal verification.

3.3 Integration with Complementary Tools

The effectiveness of TASTE and AADL within the METASAT toolchain is amplified through their integration with complementary technologies. This integration creates a continuous workflow spanning from requirements specification through to deployment and verification.

MathWorks tools form a particularly important complement to the TASTE/AADL framework. Skeletons of Simulink models can be automatically generated from the architectural specifications in TASTE, and later enhanced with behavioural details. Then, these models can be validated against formal requirements using the Simulink Requirements Toolbox, ensuring that the design meets its intended functionality.

Following validation, these models serve as the basis for code generation through MathWorks Embedded Coder and GPU Coder, producing optimised implementations for both conventional processors and specialised accelerators. This generated code then flows back into the TASTE framework, where Kazoo and Ocarina integrate it into a complete system implementation according to the AADL architectural specification.

This integrated approach delivers significant benefits throughout the development process. It maintains consistency from requirements through to implementation, enables early validation of system properties, automates routine aspects of development, and provides a framework for reasoning about complex system behaviours before committing to hardware implementation.

4 TASTE in the METASAT Workflow

The METASAT workflow demonstrates how TASTE can be systematically integrated across all the phases of the satellite system development lifecycle. This section examines the specific roles TASTE plays within this workflow and how it contributes to a more efficient and reliable development process. A more detailed description of the METASAT workflow can be found in [10]. Figure 1 shows how TASTE is integrated in the METASAT workflow.

4.1 Architectural Design and Validation

Following initial requirements specification, TASTE assumes a central role in system architecture design. Rather than treating architecture as a static blueprint, METASAT uses TASTE to create a dynamic architectural model that serves as the foundation for subsequent development phases.

The process begins with the TASTE Interface View, where engineers define the functional architecture of the system. This view captures not just component boundaries but their interactions through formally defined interfaces. The complementary TASTE Data View employs ASN.1 to specify the data structures exchanged between components, including constraints on valid values.

For physical architecture, the TASTE Deployment View creates explicit mappings between functional components and hardware resources. This mapping is particularly critical in the METASAT context, where specialized hardware accelerators must be efficiently utilized while maintaining system integrity.

Architecture validation occurs continuously throughout this process. The formal models enable automated verification of structural properties, while integration with the Simulink Requirements Toolbox allows engineers to confirm that the architecture satisfies functional requirements.

4.2 Model-Based Design and Code Generation

Once the architecture is validated, TASTE facilitates the transition to detailed design through automated model skeleton generation. The tool produces Simulink block templates that match to the interfaces defined in the architectural model, ensuring consistency between architecture and implementation.

Engineers can then enhance these skeletons with behavioural details guided by system requirements. The resulting models can include both the application logic that will execute on the satellite and simulations of the physical environment with which it interacts.

In the code generation phase, TASTE orchestrates the transformation from models to executable code. The workflow leverages MathWorks tools to generate optimised implementations, using Embedded Coder for conventional processors and GPU Coder for specialised accelerators. TASTE then integrates this generated code with the system architecture through Kazoo and Ocarina.

This integration process handles not only the functional code but also the configuration required for deployment on the target platform. This includes specialised settings for high-performance features like OpenMP, configurations for the SPARROW AI accelerator and Vortex GPU, and partition definitions for the XtratuM hypervisor.

4.3 Integration Verification

For system integration verification, The AADL models created by TASTE drive the composition of a comprehensive Digital Twin. This virtual representation integrates software artifacts from application code through operating systems and device drivers with models of the target hardware platform.

The Virtual Processor in the Loop (VPiL) approach enables testing of the actual executable code without requiring physical hardware. This capability proves invaluable for early verification of both software-to-software and hardware-to-software integration.

Throughout this process, the formal models created by TASTE provide the reference against which integration is verified. The explicit specification of component interfaces and interactions enables automated detection of integration issues, significantly reducing the manual effort required for system verification.

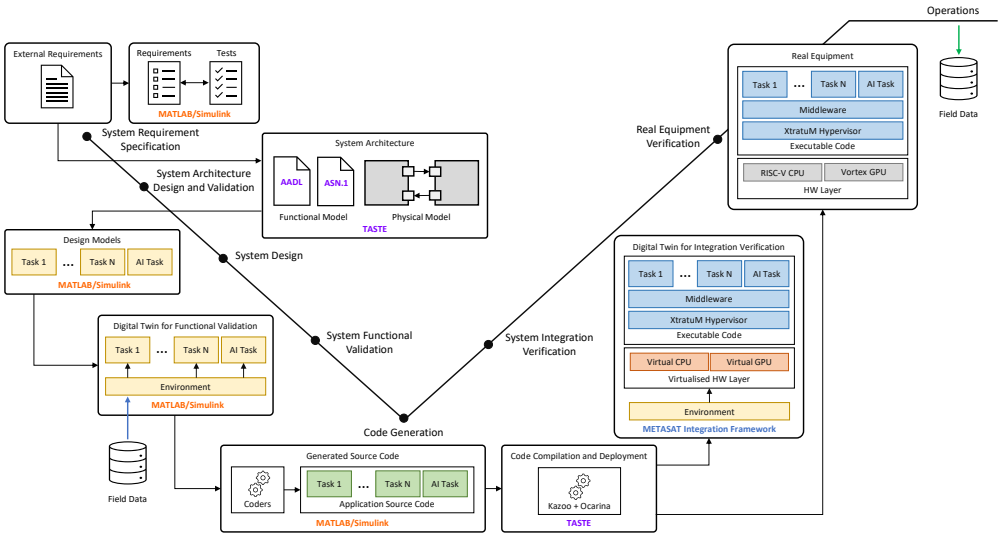


Figure 1: The METASAT workflow

5 Conclusion

The METASAT approach leverages TASTE and AADL across the phases of the development cycle, creating a continuous workflow from requirements specification to system verification. This integration delivers several key advantages that directly address the challenges facing modern satellite development. The integration between TASTE and complementary tools creates a development environment that leverages the strengths of each technology. The combination of the formal precision of TASTE with the simulation and code generation capabilities of MathWorks tools enables a development process that maintains consistency from requirements through to implementation. The METASAT approach demonstrates how TASTE and AADL can be extended to support advanced hardware architectures that incorporate heterogeneous processing elements. By providing a framework for managing the complexity of multicore processors, AI accelerators, and GPUs within a hypervisor-based environment, METASAT opens new possibilities for high-performance satellite systems that maintain the dependability required for space applications.

References

[1] L. Kosmidis, A. J. Calderón, A. Álvarez, S. Sinisi, E. Göhler, P. Gómez, A. Hönle, A. Jover, L. Lazzara, M. Masmano, P. Onaindia, T. Poggi, I. Rodríguez, M. Solé, G. Stazi, M. M. Trompouki, A. Ulisse, V. Di Valerio, J. Wolf, and I. Yarza, "METASAT: Modular Model-Based Design and Testing for Applications in Satellites," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pp. 347–362, Springer Nature Switzerland, 2023.

[2] M. Solé and L. Kosmidis, "SPARROW: A Low-Cost Hardware/Software Co-Designed SIMD Microarchitecture for AI Operations in Space Processors," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2022.

[3] B. Tine, K. P. Yalamarthy, F. Elsabbagh, and K. Hyesoon, "Vortex: Extending the RISC-V ISA for GPGPU and 3D-graphics," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 754–766, 2021.

[4] M. Perrotin, E. Conquet, P. Dissaux, T. Tsiodras, and J. Hugues, "The TASTE Toolset: Turning Human Designed Heterogeneous Systems into Computer Built Homogeneous Software," in *ERTS2 2010, Embedded Real Time Software & Systems*, 2010.

[5] M. Perrotin, E. Conquet, J. Delange, A. Schiele, and T. Tsiodras, "TASTE: A Real-Time Software Engineering Tool-Chain Overview, Status, and Future," in *SDL 2011: Integrating System and Software Modeling*, 2012.

[6] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The Architecture Analysis & Design Language (AADL): An Introduction." Carnegie Mellon University, Software Engineering Institute, 2006.

[7] International Telecomunication Union, "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation," *International Standard X.680*, 2002.

[8] TASTE, "Kazoo." `https://taste.tuxfamily.org/wiki/index.php?title=Kazoo`. Accessed April 2025.

[9] J. Hugues and J. Delange, "Model-Based Design and Automated Validation of ARINC653 Architectures Using the AADL," in *Cyber-Physical System Design from an Architecture Analysis Viewpoint: Communications of NII Shonan Meetings*, pp. 33–52, Springer, 2017.

[10] A. J. Calderón, I. Yarza, S. Sinisi, L. Lazzara, V. Di Valerio, G. Stazi, L. Kosmidis, M. M. Trompouki, A. Ulisse, A. Amonarriz, and P. Onaindia, "The METASAT Model-Based Engineering Workflow and Digital Twin Concept," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2024.

Executable AADL Models for Early System Qualification Test

Stéphane Rubini, Sébastien Levieux, Eric Cariou, Frank Singhoff, Hai Nam Tran
Univ Brest, CNRS, Lab-STICC, Brest, France; email: {firstname.lastname}@univ-brest.fr
Gilles Le Pluart
THALES Defense Mission Systems France, France; email: gilles.lepluart@fr.thalesgroup.com

Abstract

This paper presents an approach, called CosiCosi*, whose the aim is to characterize the temporal behavior of complex systems. The analyzed systems are defined by their software and hardware architectures, and by the duration of the processing jobs. The abstraction of jobs by their execution time enables early verification, at a phase where all the implementation details of the components are not yet defined.*

The tool that supports the verification is a SystemC simulator configured from AADL *architecture models. It simulates the temporal behavior of the model components, respecting the* AADL *runtime specification as defined in the standard.*

The targeted use of the tool is the qualification of hardware/software architectures through intensive testing under an expected workload. The evaluation shows the tool's ability to simulate a system composed of 600 threads, over 1 hour of operation, in approximately 2 minutes on a desktop computer.

1 Introduction

The qualification tests are performed on complete systems, including software and hardware components. Their objective is to check that the system is compliant with the system requirements. However, discovering problems at this phase of the life cycle is known to be costly and time consuming [1].

In the context of model driven engineering of real-time systems, we advocate that a preliminary phase of this qualification testing activity can be realized at early stages of the designing process. The simulation arrives as a mean for reinforcing design choices during the elaboration of the system architecture. Obviously, the entire code is unknown at this phase time, and test must be done from abstraction like the budgeted execution time or the excepted rate of data exchanged between components.

The systems we consider are too complex, and heterogeneous, to fulfill the conditions that allow for static real-time scheduling analysis methods to be applied. Moreover, the critical instant, – the instant where the system is in the most critical state for fulfilling its timing constraints –, is unknown. It is not even sure that the operations performed on the system ever lead to that critical state. Complex systems are also characterized by the risk that emerging behavior appears. These emerging behaviors are difficult to predict from the separated analysis of the different parts of system.

Hence we need for a simulation tool to test a significant part of reachable states of the entire system. The functionality of the tool are outlined below:

- Load and performance testing to validate the system temporal behavior under an operational activity;
- Generation of test inputs according to this operational activity, through the modeling of the requests (by algorithms, traces, statistic laws ...);
- Definition of monitoring points inside the system, and request tracking to support performance testing.
- Reproducibility of the simulation in order to analyze the sequence of operations that leads to a given state.

The model of the system to qualify is expressed with the language AADL. AADL allows us to describe the architecture of the system from the hardware and software points of view. Component properties supply the information to characterize the temporal behavior of the system components.

CosiCosi, a new tool we are developing, is a SystemC [2] simulator configured from AADL system architectures. It is generated by model transformation from this architectural model. The generated simulator must take into account the application software, but also the runtime of the system through its operating system and hardware execution platform.

In the sequel of this paper, we outline the AADL/SystemC transformation process. Next, we present the structure the simulator and the major aspects of its implementation. The last part shows a simulation example and evaluates the scalability of the approach. We conclude by describing the current status of the project.

2 The CosiCosi Simulator

Figure 1 outlines the different layers to consider to generate the *CosiCosi* simulator of an AADL model.

The AADL threads and data components constitute the software layer that is transposed into the simulation models. The

behavior of the threads is abstracted by their scheduling properties and by pseudo-instructions that represent computation time between two operations that could affect the scheduling (for instance two sends of value).

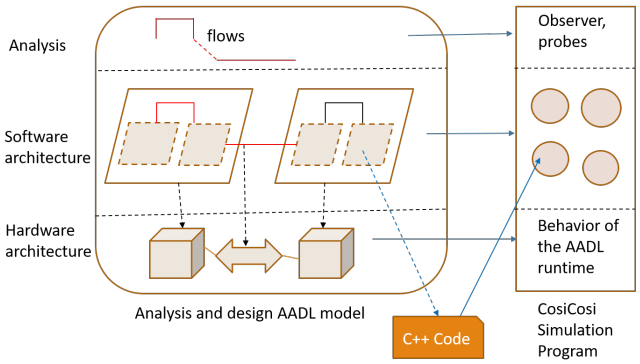


Figure 1: AADL/SystemC transformation.

The major element considered in the hardware architecture is the processor component. AADL defines processor as "an abstraction of hardware and software that is responsible for scheduling and executing threads". Representing processors is mandatory to simulate the timing behavior of the system.

Finally, the C++ codes referenced in the AADL model is intended to model the requests the system must proceed following the application requirements.

2.1 AADL/SystemC Transformation Process

The transformation AADL/SystemC could have be built from an equivalence between the AADL thread component and the SystemC THREAD concept, both representing activities of the system. However, a major discrepancy between the two "worlds" is that AADL software threads need an access to an execution support to progress (i.e. a processor) whereas a SystemC thread models a "physical" activity of a part of the system, parallel by nature with the activity of the other parts.

Then, the simulation model must represent the sharing rules of the hardware execution platform between the software threads. In the AADL standard, these rules are encompassed in the AADL component semantics, and configured by the means of user-defined properties. To handle the AADL runtime support, which is implicit in the AADL models, we are implementing the library AADL_RT, based on a modular SystemC RTOS simulator called SysRT [3].

Figure 2 outlines the way AADL components are translated in the CosiCosi simulation model. The *sc_module* object represents a component, i.e. a structural element characterized by an interface, an internal state and a behavior. The *sc_port* objects model the input/output ports of the modules, and implement a C++ interface that allows connection to *sc_channel* objects. This last class is a base class to realize the AADL connections.

Then, the structural aspect of the system is captured by basic features of the SystemC library. In addition, OS modeling is supported by the services of SysRT, which simulate tasks, task scheduling, CPU and resource usages.

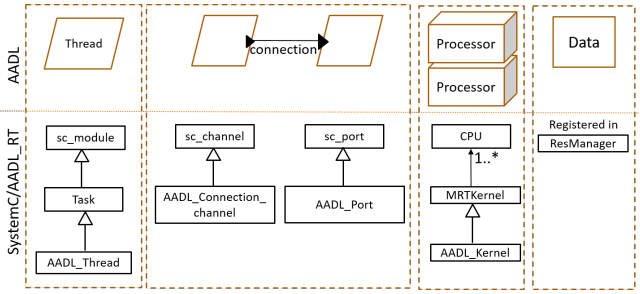


Figure 2: Simplified transformation rules. The "sc_" prefix stands for SystemC classes, and "AADL_" for AADL_RT classes. The others classes come from the SysRT library.

3 Implementation Details

In the section, we detail some key aspects of the implementation of the CosisCosi simulator: The main principles of SysRT library, the configuration of the AADL component models through their properties, and the implementation of the AADL connection.

SysRT fundamental SysRT allows for an event driven simulation of a real-time system running on a kernel that implemented preemptive scheduling policies. The times at which the simulated system state is updated are calculated from the intended scheduling events. So, the role of SysRT objects is the production of different types of events that will describe the future instants when the state of the RTOS must be update (for instance arrival of a new job, a job deadline, a scheduler invocation...). Pseudo instructions represent the duration of the execution of a task, or operations that may change the scheduling of the system (value send, shared resource accesses...). .

AADL property database An internal database is populated by all the properties issued from the AADL model. During the elaboration phase, each component instance is in charge of extracting the properties that concern it from the database, and configures its behavior following the semantic of these properties (see Figure 3).

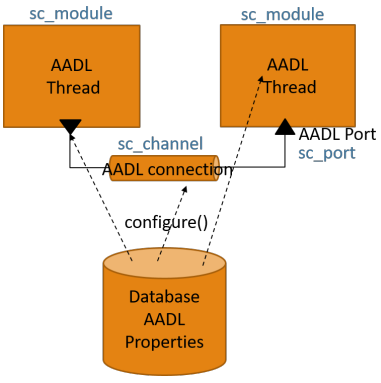


Figure 3: Property database.

This approach allows us to simplify the model transformation process and the SystemC component constructors. As the

SystemC classes that implement AADL components are responsible for reading and interpreting the properties relevant to them, new properties can be taken into account at the simulator level without any updates in the model transformation process.

Furthermore, some SystemC entities, like the *sc_channel* access ports, become reachable at the end of elaboration process (i.e. no communication are allowed on connections before all ports are connected), and then all the port configurations must be triggered at the end of the elaboration.

Communication channel The AADL standard allows for a N-to-N connectivity on a connection channel. The AADL connection semantic raises some issues for its implementation in the SystemC channel model.

- The connection may receive multiple incoming events or data events from different source ports at the same time. Then, a entry queue, the "mux queue", is necessary for buffering them in the channel.
- A data or event may be duplicated in order to be transferred to all input ports connected to the channel. Each input ports may consume the data/event at different times. So, a SystemC thread broadcasts the event/data to all the input ports where queues store them temporally before the threads use them.
- If the input ports of a given thread is bound to different connections, the event/data consumed at the next dispatch of the thread is defined by the global arrival order of the data/events. In the SystemC implementation of the channel, a queue maintains this global order, and points to the port that will provide the next data/events.

Fig. 4 illustrates this architecture.

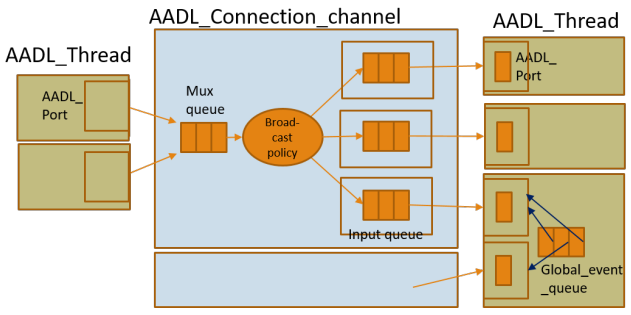


Figure 4: CosiCosi implementation of AADL connection.

4 Evaluation and Project Status

In this section, we illustrate the results the simulator can produce through a demonstration project that include several AADL port types and thread dispatch protocols. We then evaluate the scalability of the simulator for long simulation times et large systems.

Name	Period	Capacity	Dispatch	CPU
sensor0	20ms	2ms	Periodic	c0
sensor1	30ms	2ms	Periodic	c0
sensor2	40ms	2ms	Periodic	c0
thr_collector	8ms	5ms	Sporadic	c0
thr_savestat	-	5ms	Aperiodic	c0
thr_reqstat	50ms	$[2ms, 49ms]$	Periodic	env

Table 1: Parameters of the 6 threads in the example.

4.1 Evaluation

Example: Checking the depth of a message queue In that example, we investigate whether the depth of a FIFO file is large enough to avoid data loss. Data from three sensors are collected by 3 threads at different rates and send through a communication channel to another thread in charge of maintaining statistics. The statistics, stored in a shared memory, may be accessed at any time by 2 threads (i.e. *thr_savestat* and *thr_collector*), but only in mutual exclusion. Figure 5 shows the system architecture.

Another thread *thr_reqstat*, not represented in the figure, is in charge to generate requests that trigger the *thr_savestat* thread through the event port "r". It models the usage of the system by an external actor. Inter-arrival time of the requests is in the interval $[3ms, 48ms]$.

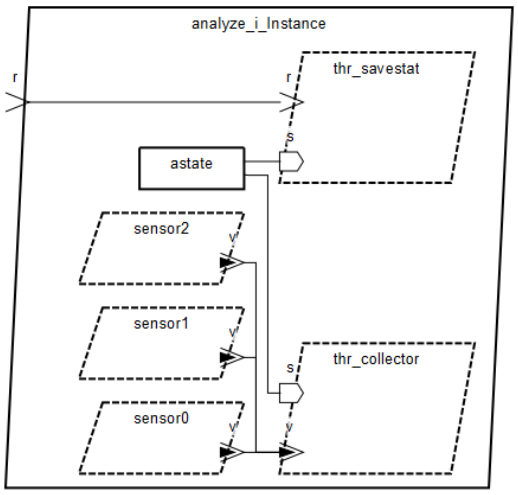


Figure 5: First system example. As shown by the AADL diagram, the process "analyze" needs for its implementation event data ports "v" to communicate the sensor values, data port for accessing the shared data "aState", and an event port "r" to trigger the thread "thr_savestat". "sensor0", "sensor1" and "sensor2" are *periodic* threads, released at different rates, "thr_collector" is a *sporadic* thread triggered by the arrival of sensor data. "thr_savestat" is *aperiodic*, dispatched by event produced by an external requester.

The designer aims to check the impact of the statistic result accesses on the filling of the communication channel FIFO, and then the rate of requests that can be allowed without sensor data loss. On Figure 6, which shows a part of a simulation run, the "signal" `col_queue_size` varies from 0 to 2; This signal represents the filling of the communication queue between the sensors and the collector thread. As the

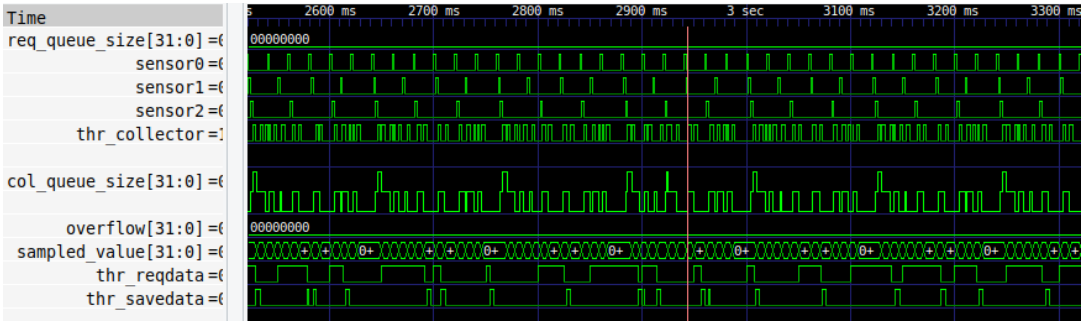


Figure 6: Simulation Trace Presented by the Tool *gtkwave*.

queue depth is set to 3 in the system model, the simulation did not exhibit queue overflow, i.e. data loss, over the specified request rate.

Scalability These experiments aim to check the ability of the tool to handle complex systems during long simulation times. This feature is important with regard to the expected usage of the tool.

The first set of measures is applied to the previous example (6 threads, 2 connections, 2 CPUs, 10 traced features) by varying the system time interval[1]. Figure 7 shows the simulation time for system time intervals as long as 10 hours (36000 s).

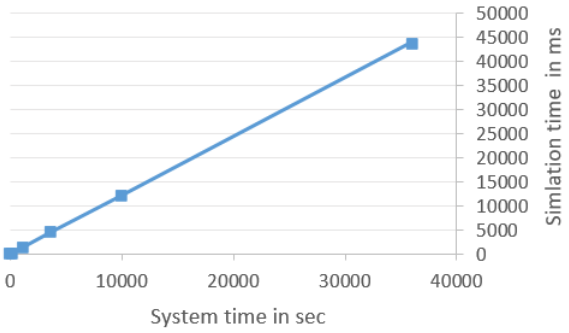


Figure 7: Simulation speed.

The simulation time increases linearly with the system time (around 1 ms to simulate 1 s of the system time). The trace file (VCD format) contents around 3 millions of events for a simulation of 10 hours (system time). The viewer VCD GtkWave is able to present “smoothly” this amount of events.

The curve of Figure 8 shows the number of events traced per ms for the same experiment. As soon as the duration of the simulation becomes dominant compared to that of the elaboration phase, the tracing speed reaches an almost constant rate of around 650 events/ms.

The next experiment checks the ability of the CosiCosi simulator to deal with large systems. We build such systems by duplicating the example system several times. The largest system we tested, during 1 hour of system time, contains 600 threads, 200 CPUs and 200 connections.

[1]We use the term "system time" for representing the time at the level of the simulated system, and "simulation time" for the execution time of the simulator itself.

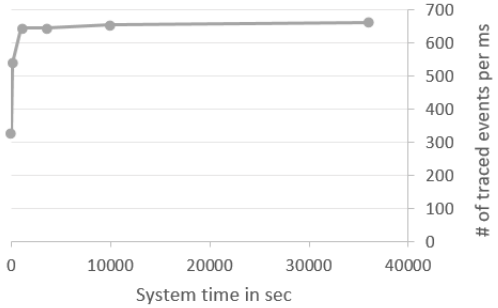


Figure 8: Number of traced events per ms.

As shown in Figure 9, the simulation time increases linearly with the thread number (around 200 ms per thread, with a slight upward inflection for the last measure) The trace file contents around 80 millions of events for a simulation of 600 threads on 200 CPUs.

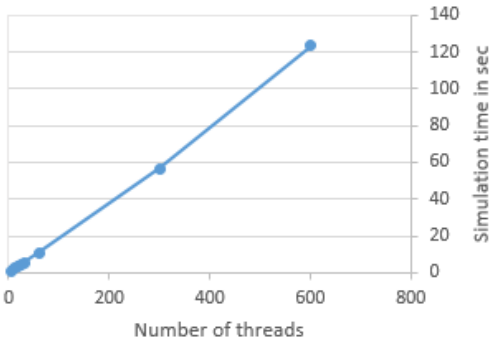


Figure 9: Effect of the number of threads on the simulation time. The system time is 1 hour.

Project status The project was initiated for the need of a case study provided by the company *Thales DMS*: The system to qualify in this case is a wide distributed application that must meet real-time constraints. Some components of the system represent classical compute nodes or software threads, whereas others are for an abstract temporal modeling of complex subsystems. The objective is to check the respect of the real-time constraints, and to evaluate the load of some entities of the system in order to validate design choices.

The work presents in this paper is an in-progress work. We

are currently working on the implementation of new features in the library AADL_RT. The model transformation is carried on the OSATE framework. An intermediate DSL (Domain Specific Language) was defined to model the concepts needed for generating the target temporal simulator. The first step of the transformation is adapted from the work presented in [4].

5 Related Works

Several tools work from AADL models to drive different kinds of analysis or code generation processes. For instance, the tools *cheddar* realizes scheduling analysis, *marzhin* (AADL *Inspector*) [5] simulates the model behavior for statically defined scenarios, or *ocarina* produces executable application by assembling the functions referenced in the model.

In [6], the authors provide a more detailed description of hardware components than that allowed by AADL properties. A state machine defines the internal behavior of each component, and supports variation in its temporal behavior. An abstract action and cost are associated to each state. An AADL model may reference a given automata in a database through an AADL property value.

The tool AADS [7] transforms an AADL model into an executable model in SystemC. AADL components are implemented by library functions in C++/C; For instance an ADDL thread is transformed into a POSIX thread, a `Compute_Execution_Time` property into a program call consuming the specified calculation time. The produced executable model then relies on the services of the SCoPE library, which extends SystemC to co-simulate a virtual hardware platform and C++ software codes.

Unlike these two approaches which deal more specifically with the hardware part of the AADL model, the CosiCosi proposal includes the modeling of the AADL runtime in SystemC.

The AADL-DEVS [8] framework proposal allows for dynamic simulations of AADL models. DEVS formalism is an extension to Finite State Automata which provides a support for discrete-event simulation and continuous-time modeling. The AADL system architecture can be enriched with DEVS models that specify thread behaviors, and simulated with the DEVS-Suite discrete event simulator after a model transformation. But this framework does not seem to take into account the hardware architecture of the system.

Conclusion

This article presents the *CosiCosi* approach. The problem we are addressing is the qualification test of the architecture of large distributed systems, and this as early as possible in the development process. The SystemC implementation provides the flexibility to express the application workload as C++ programs.

Despite this work is in progress, the current implementation of AADL_RT was used to simulate an AADL model that encompass several AADL concepts, and has demonstrated the potential of the simulator to deal with large systems.

In the context of intensive tests, the assess of some system properties must be automatically done. We will study how virtual probes could be inserted and configured in the AADL model, to express the properties to monitor during the test.

Acknowledgment This work is carried out with the support of the Cormorant Scientific Interest Group and Thales DMS. The authors thank Dominique Blouin for his help on the AADL/SystemC transformation, and Jun Xiao for the SysRT library.

References

[1] G. Tassey, "The economic impacts of inadequate infrastructure for software testing," tech. rep., National Institute of Standards and Technology (NIST), 2002.

[2] "IEEE standard for standard systemc® language reference manual," *IEEE Std 1666-2023 (Revision of IEEE Std 1666-2011)*, pp. 1–618, 2023.

[3] J. Xiao, A. Pimentel, and G. Lipari, "SysRT: A modular multiprocessor RTOS simulator for early design space exploration," in *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pp. 38–45, IEEE, 2017.

[4] P. Bomel, D. Blouin, M. Lanoe, and E. Senn, "Functional validation of aadl models via model transformation to systemc with ATL," in *Proceedings of the 5th International Workshop on Model Based Architecting and Construction of Embedded Systems*, pp. 13–18, 2012.

[5] P. Dissaux and O. Marc, "Executable aadl," in *ACVI 2014–Architecture Centric Virtual Integration Workshop Proceedings*, p. 59, Citeseer, 2014.

[6] M. Lafaye, L. Pautet, E. Borde, M. Gatti, and D. Faura, "Model driven resource usage simulation for critical embedded systems," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 312–315, IEEE, 2012.

[7] R. Varona-Gómez, E. Villar, A. I. Rodríguez, F. Ferrero, and E. Alaña, "Architectural optimization & design of embedded systems based on aadl performance analysis," *American Journal of Computer Architecture*, vol. 1, no. 2, pp. 21–36, 2012.

[8] E. Ahmad and H. S. Sarjoughian, "An environment for developing simulatable aadl-devs models," *Simulation Modelling Practice and Theory*, vol. 123, p. 102690, 2023.

AADL Modeling and Schedulability Analysis of Multiprocessor Architectures

Frank Singhoff, Stéphane Rubini, Hai Nam Tran, Sébastien Levieux
University of Brest, Lab-STICC UMR CNRS 6285, Brest, France; email: {firstname.lastname}@univ-brest.fr

Abstract

Although significant efforts has been made to investigate schedulability analysis in the context of multiprocessor real-time systems, performing such an analysis stays difficult to achieve in practice. In this article, we present an approach to model and perform schedulability analysis with the Architecture Analysis and Description Language (AADL) and its ecosystem. We show how to model with AADL different types of multiprocessors and how to assess schedulability. These proposals have been implemented as an OSATE plug-in and in the Cheddar tool.

1 Introduction

Various kinds of multiprocessor have been designed during the last decades. For example, multicore processors have emerged in the early 2000s when it became clear for the computing system designers that the cost of further increase in unicore processor performance will be too expensive. The extremely high design complexity of and implantation issues such as hot spots at some points on the chip, hinders the potential gain in this way. Polack's rule [1] states that the performance efficiency per transistor and per watt tends to decrease when the single core size increases.

Multicore processors produce computing power through the association of multiples processing cores. These cores are simpler, and based on less aggressive design than the last generation of unicore processors. The aim of multicore chips is to reduce Size, Weight and Power (SWaP) requirements in the implementation of intensive computation systems.

Complex embedded systems, such as those used in modern transport vehicles, are composed of multiple functions. Beside the critical functions, applications with low criticality, but high performance requirements have to be embedded. The number of such kind of applications is expected to increase in the future. Since computations in embedded systems are inherently parallel, they are well-adapted for implementation on highly parallel platforms, especially multicore and many-core processors. By grouping the execution of functions on a single die, the systems are globally more integrated. The expected result is a decrease of the cost, and a better reliability by limiting interconnection hardware, which is known as a source of failures (30% in the case of automotive [2]).

To summarize, various kinds of multiprocessor architectures exist, depending on the properties of the processor units, the way they are organized/connected, or the way software are deployed on.

Problem statement: When stakeholders perform early verification for certification purposes for example, efficient verification methods exist for uniprocessor platforms. Unfortunately, while 80% of stakeholders use multiprocessor platforms, the verification of such architecture stays an active research track [3].

Contributions: In this article, from a classification of multiprocessor architectures, we propose guidelines to model multiprocessor architectures with AADL [4]. We present design patterns and new AADL properties for such purpose. We also introduce schedulability analysis methods implemented in Cheddar tool [5] which are able to analyze AADL models compliant with the proposed guidelines. This article contributes to understand what stakeholders have to model or abstract when designing their AADL models with schedulability in mind. It also contributes to identify what are the missing properties for schedulability analysis in the current AADL standard. The resulting models can be analyzed by simulation to access schedulability, providing systems are sustainable [6] and that a feasibility interval exists [7]. These proposals have been implemented as an OSATE [8] plug-in and in the Cheddar tool.

This article is organized as follows. Section 2 introduces a classification of multiprocessor architectures. Such a classification has been used to propose AADL modeling guidelines introduced in Section 3. Schedulability analysis methods available in Cheddar for AADL models are shortly introduced in Section 4. Finally, the article is completed with a short presentation of related works in Section 5 and a conclusion in Section 6.

2 Multiprocessor Classification

The fundamental aspect of the processor architectures addressed in this article is the ability, at the hardware level, to run several programs at the same time, and consequently to manage simultaneously several hardware execution contexts. A hardware context typically includes program counter, general purpose bank registers, and state registers.

The number of available execution contexts is a measure of the maximum level of parallelism of an architecture, even if various resource access conflicts limit the actual number of programs that run simultaneously. The identification of the

shared resources is a major aspect in the description of multi-processing systems, as they play a key role in the performance and the predictability of the system. Figure 1 illustrates the major, although not mutually exclusive, solutions: multi-processors, multi-cores and multi-threading.

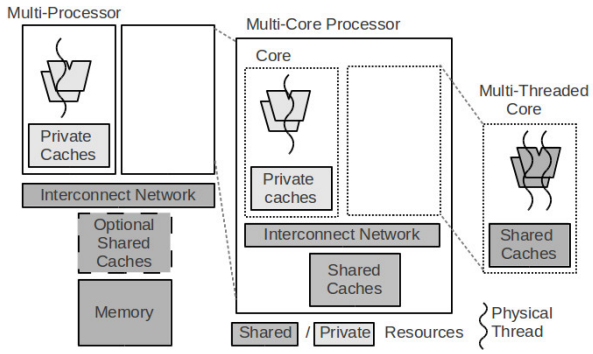


Figure 1: Architectural localization of the multi-processing capabilities. The ability of executing multiple threads may be provided at different levels: by the coupling of processor chips, inside a processor chip or inside a data-path.

Historically, multiprocessor architectures have been the first solution. The processors are located on different integrated circuits, and share a memory. Memory and bus contention are major concerns in this kind of architecture, and the off-chip implementation of the buses limits the field of implementations that could be used to deal with this problem. The off-chip buses are also known to consume a significant part of overall power budget [9].

Today's multicore processors are only integrated implementations of multiprocessor systems. However, the technical constraints are not the same, and more efficient solutions may be developed to interconnect the cores and the memory hierarchy inside a chip. Multiplexed bus, crossbar switches or Network-On-Chip [10] are examples of interconnect architectures that allow parallel transactions between cores, memories, or I/O devices.

To summarize, architectures presented above may cover various types of concurrency. In this article, we propose AADL modeling guidelines that allow designers to model a broad class of multiprocessor architectures for the purpose of schedulability analysis. In the sequel, we classify multiprocessor architectures according to three criteria: the model of architectures, the ability to have multithreading or not, and the kind of operating systems to operate the hardware resources.

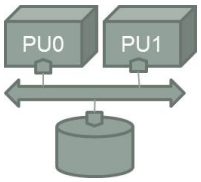
2.1 Architecture Models

Table 2 summarizes properties of the three architecture models addressed in this article. Each model is composed of processing units and means allowing these units to interact. Computing units express concurrency limited by their interactions. Interaction can be achieved either by shared memory of by network communications. The order of magnitude and the predictability of interaction latencies are different between shared memory and network communications, which is illustrated by the three models we consider.

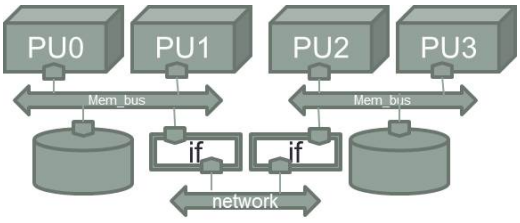
Programming Paradigm	Comm. Model	Software Support	Hardware Support
Distributed	Message	Native	Network-On-Chip
Distributed	Message	Library fcts	Memory hierarchy
Shared memory (UMA)	Variable	Native (OpenMP)	Memory hierarchy
Shared memory (NUMA)	Variable	Native (OpenMP)	Memory hierarchy + Shared memory + Network-On-Chip

Figure 2: Main properties of architecture models considered in this article

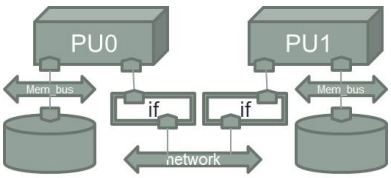
Figure 3a introduces the UMA (Uniform Memory Access) model. In UMA architectures, computing units share the same physical memory unit, i.e. then, memory accesses have similar duration for any computing units. In contrary with the NUMA architecture shown in Figure 3b, memory access durations vary between processing units as each computing unit has its own private memory. Interaction latencies and predictability also depend on the devices and protocols used to implement network communications. In the last model shown in figure 3c, each processing units has its own (private) memory and interactions between them are only implemented by network communications.



(a) Uniform Access Memory model



(b) Non Uniform Access Memory model



(c) Distributed (message passing) model

Figure 3: Architecture Models

2.2 Multi-threading

Another source of concurrency that we intend to model is multithreading [11]. Multithreading strategies have been introduced to increase the usage efficiency of processing units inside the pipeline of superscalar processors. When the instruction level parallelism (ILP) is too limited and/or the execution flows are stalled, waiting for the end of memory

requests, processing units are available for executing instructions issued from other threads.

However, the concept of multithreading inherently leads to the sharing of processing units within a core or processor. In the general case, the starting up of an additional thread impacts the performances of other threads running on the same processor. This implicit interference between threads[1] is very difficult to quantify, because it depends on the set of instructions, coming from the candidate threads, that can be issued at a given time towards the processor pipelines.

2.3 Operating Systems

Architectures described above have to host an operating system to manage resource allocations. In [12], two approaches have been proposed for such a purpose:

- Asymmetric Multiprocessing (AMP): each core runs a separate operating system, which only manages its own core. The whole system looks like a distributed one, with communication channels created in the application code. Operating systems hosted by each core may be different or not.
- Symmetric Multiprocessing (SMP): several cores are controlled by a unique operating system. The operating system knows the parallel nature of the system, and its internal structure is designed to work safely in that environment.

AMP and SMP models have both their own advantages and disadvantages; In the context of avionic systems, the reader can refer for instance to [13].

3 AADL Modeling of Multiprocessor Architectures

Most of the works addressed by the AADL community focused on partitioned architectures modeling and verification. In this article, we aim to model a broader class of multiprocessor architectures, including architectures with global scheduling, task migrations and interference due to hardware components.

When modeling a multiprocessor platform, we do not intend to model hardware in detail. We restrict the modeling to entities that are mandatory for schedulability analysis. As stated in [14], we intend to model any entity of a multiprocessor platform that implies concurrency or leads to task interference. Then, we model processing units, cache units, interconnect units, and any of their properties useful for schedulability analysis.

3.1 Modeling Guidelines

We propose to model a processing unit by an AADL `system` component. This component may contain AADL processor components to express concurrency sources, or more generally, to model any resource sharing/scheduling. Then, a `system` component may define subcomponents to model resource contentions.

We assume that the following guidelines are applied to model a multiprocessor platform:

1. In the AADL standard, an AADL `processor` is the execution platform component that is able to schedule and execute threads. An AADL processor is able to execute zero or one sequential flow of control at a time, and to store the architectural state associated with this execution. The execution flow of a processor is referred later as a "physical thread". A sequential control flow, i.e. a physical thread, is then modeled by an AADL processor.

2. An AADL processor models a physical thread, without considering its hardware implementation. Indeed, the hardware supporting the execution of the flow may be implemented by different means.

3. If a processing unit contains several physical threads, they are grouped in the same AADL `system` component.

4. Any `system` component modeling a processing unit or one of its subcomponents must define the property `Supported_Soc_Type` to specify the semantics of the component. Property `Supported_Soc_Type` is defined as follows:

   ```
   Supported_Soc_Type : type enumeration (
     SoC_Processing_Unit, SoC_Memory_Unit,
     Soc_Interconnection_Unit );
   System_Soc_Type : Supported_SoC_Type
     applies to (system);
   ```

   Hence, a `system` component modeling a processing unit must define the `Supported_Soc_Type` property with the value `SoC_Processing_Unit`.

5. A `system` component modeling a processing unit may define subcomponents (e.g. thread, data ...) to model resource contention in the processing unit that may introduce potential interference on the application. Subcomponents can be any AADL components, including `system` components modeling memory units or interconnection devices. In that case, the property `Supported_Soc_Type` is set with respectively the values `SoC_Memory_Unit` and `Soc_Interconnection_Unit`.

Figure 4 is an example of a dual-core processor modeled by a `system` component. Notice property `System_Soc_Type` which specifies that the `system` component is modeling a processing unit. Notice also property `SoC_Interconnection_Type` which specifies how cores are connected. In this example, cores are connected to the memory unit by a shared bus, which leads to potential interference on memory accesses, interference that have to be considered during the schedulability analysis.

In the sequel, we address the modeling of the application deployment, how to express the processing unit performances and the hardware shared resources.

[1] The same problem exists when threads share other resources like memory buses for instance.

```
-- A single physical thread
processor uni_core
properties
  Scheduling_Protocol=>(RATE_MONOTONIC_PROTOCOL);
end uni_core;

system dual_core end dual_core;
system implementation dual_core.impl
subcomponents
-- Two physicals threads
  core1 : processor uni_core;
  core2 : processor uni_core;
properties
  System_Soc_Type=>SoC_Processing_Unit;
  SoC_Interconnection_Type=>Shared_Bus;
end dual_core.impl;
```

Figure 4: AADL model for of a dual-core processor.

3.2 Deployment of Applications on Processing Units

AADL represents the deployment of an application by binding properties, and hence associates software entities to execution resources. For instance, the following property assigns the execution of a thread ta, belonging to the process $as1$, to a processor $p1$:

```
Actual_Processor_Binding=>(reference(p1))
  applies to as1.ta;
```

Now, we explain how this binding mechanism may be used in the multi-processing context.

Partitioned scheduling To address partitioned scheduling, a list of software threads may be assigned to their targeted processors by duplicating the processor bindings for all of them. As AADL defines a processor as an executing platform including a scheduler that manages the sharing of its computing resources, the system is modeled as a set of nodes with their associated local scheduling policy.

Remind that one AADL processor is associated to one physical thread, i.e. hardware executing one flow of control, regardless whether this processing unit constitutes a core of a processor, or a processor as a whole (we postpone the case of multi-threading later in this section). Hence, an AADL processor includes the private resources dedicated to a flow of control, mainly the execution pipelines and the private cache memories (see figure 1). Such a modeling guideline aims to exclude of the processing unit all the shared resources between them, in order to define independent subsystems when they do not need to interact through external resources. Shared resources are often a bottleneck in multiprocessors. The models should highlight them (see at the end of this section).

Global scheduling A global scheduler maintains a system-wide queue of ready tasks and considers a set of processing units to execute them. An AADL processor deals with a single hardware execution flow at a given time, and cannot be used to model a set of processing units. The approach is to group the target processing units into a system component, and to bind the threads to this multiprocessing system (smp in the example below):

```
Actual_Processor_Binding=>(reference(smp))
  applies to as1.ta;
```

An alternative binding method enumerates explicitly the set of processing units involved in the scheduling:

```
Actual_Processor_Binding=>
  (reference(smp.p1),reference(smp.p2))
    applies to as1.ta;
```

An AADL processor includes a scheduler, of which the type is specified by the standard property `Scheduling_Protocol`. In the case of global scheduling, the value of this property must be the same for all the AADL processors. To remove this consistency rule, the AADL standard could be adapted to associate a global scheduling method to all the processing units available within the corresponding system component. As shown in Figure 5, new properties have been also proposed to customize global scheduling policies.

```
-- Specify the kind of migration allowed by the
-- scheduling policy
Supported_Migrations_Type : type enumeration (
  No_Migration_Type, Job_Level_Migration_Type,
  Time_Unit_Migration_Type);
Migrations_Type : Supported_Migrations_Type
  applies to (memory, system);

-- Specify if processing units of an AADL processor
-- are homogeneous or not
Supported_Multiprocessors_Type : type enumeration (
   Identical , Homogeneous, Heterogeneous);
Multiprocessors_Type : Supported_Multiprocessors_Type
  applies to (memory, system);
```

Figure 5: Properties for global scheduling

3.3 Expressing Processing Unit Performances

In the AADL standard, timing behavior is associated to software entities by component properties. Some of them are related to the specification of the system, like the release period of the jobs of a task, but others depend on the deployment target. For instance, the execution time of a job, defined by the `Compute_Execution_Time` property may change following the processor chosen for executing the job.

AADL deals with this problem through two modeling mechanisms: (1) all the deployment dependent times are specified for a given execution target, and a scaling factor must be applied on these times when the software is bound to another target, (2) all the times are enumerated for all the possible targets.

In the first solution, the software or system components identify a reference processor, through the property `Reference_Processor`. The model of the other types of processors defines the `Scaling_Factor` property to specifies their relative speed with respect to whose of the specified reference processor.

The second solution associates a list of values for a timing property, and the component binding selects the actual value

of the property. The keyword *in binding* allows to express these conditioned definitions.

Figures 7 and 6 illustrate the two alternatives in order to model respectively a global scheduling on two uniform processors and uniprocessor DVFS (Dynamic Voltage and Frequency Scaling) capabilities.

```
system app end app;
system implementation app.i
subcomponents
  soft : process as.i;
  cpu1 : processor isa.i1;
  cpu2 : processor isa.i2;
properties
  Actual_processor_binding =>
        (reference(cpu1), reference(cpu2))
        applies to soft;
  Scheduling_Protocol=>RATE_MONOTONIC_PROTOCOL;
end app.i;

processor isa end isa;
processor implementation isa.i1
end isa.i1;
processor implementation isa.i2
end isa.i2;

process as end as;
process implementation as.i
subcomponents
  t1 : thread atask {
    Dispatch_Protocol=>Periodic;
    Compute_Execution_Time=>2 ms .. 2 ms
      in binding (isa.i1);
    Compute_Execution_Time=>3 ms .. 3 ms
      in binding (isa.i2);
    Deadline=>20 ms;
    Period=>20 ms;
      };
  t2 : thread atask { ... };
  t3 : thread atask { ... };
end as.i;
thread atask end atask;
```

Figure 6: Modeling processor speed with the `in binding` statement. The system includes two uniform processors that can be allocated to threads in a global scheduling policy. The execution time of the threads are defined according to the binding of the thread to a processor. The `in binding` statements select the value to associate to the `Compute_Execution_Time` property.

Multi-threading Multi-threaded processors or cores allow the parallel execution of few tasks [11].

A pessimistic performance scaling factor may be defined, according to the maximal number of physical threads per core, in order to take into account the performance provided by a physical thread in relation with the unithread core performance. The AADL standard property `Scaling_Factor` and `Reference_Processor` allow to specify the scaling factor with respect to a reference processor, and can be used in that goal.

Discussion Express the performance of a processor by a single "synthetic" value is of little interest for leading performance analysis. For instance, the usual MIPS parameter depends on the number of instructions that are needed to code a given function; two processors that execute a function in the same duration may have different MIPS parameters following their instruction set architecture.

```
system dvfs end dvfs;
system implementation dvfs.i
subcomponents
  soft : process as.i;
  cpu : processor isa.i;
  ctrl : process monitor;
connections
  fs: port ctrl.change_frequency->cpu.change_frequency;
properties
  Actual_Processor_Binding=>reference(cpu)
    applies to soft.t1;
  Reference_Processor<=classifier(isa);
end dvfs.i;

process monitor -- manages the processor frequency
features
  change_frequency : out event port;
end monitor;
process as end as;
process implementation as.i
subcomponents
  t1 : subprogram atask {
    Compute_Execution_Time=>10 ms .. 10 ms; };
end as.i;
subprogram fct end fct;

processor isa
features
  change_frequency : in event port;
end isa;
processor implementation isa.i
modes
  full_rate : initial mode;
  half_rate : mode;
  full_rate -[change_frequency]-> half_rate;
  half_rate -[change_frequency]-> full_rate;
properties
  Scaling_Factor=>0.7 in modes (half_rate);
  Scaling_Factor=>1 in modes ( full_rate );
end isa.i;
```

Figure 7: Modeling processor speed with the `Scaling_Factor` property. The processor accepts frequency scaling, and can run at full or half frequency (mode `full_rate` and `half_rate`). The subprogram execution time is defined with respect to a reference processor, as stated by the property `Reference_Processor` inherited from the root system component. The `Scaling_Factor` expresses that implementation `isa.i` runs at 70% of the nominal speed when half frequency mode is on.

In addition, processors do not show the same capabilities to execute different kinds of instructions. As an example, MIPS values change with the ratio between arithmetic and memory access instructions within the code of a function.

Currently, AADL associates execution time to the software entities. This solution allows to deal with the variability of the processor performance according to the program it executes. Different execution times can be defined through `in binding` statements when the software entity is allocated on heterogeneous processors. However, the `Scaling_Factor` property is globally applied on all processor operations, which

probably results in poor estimations of the execution time of the programs.

3.4 Modeling Hardware Shared Resources

In multiprocessor architectures, interference does not come only from tasks or system services, but also from shared hardware resources. Two shared hardware resources are considered: memory hierarchy units and interconnections.

```
-- Specify the types of cache
Supported_Cache_Type : type enumeration (
  Data_Cache,Instruction_Cache, Unified_Cache);
Cache_Type : Supported_Cache_Type
  applies to (memory, system);

-- Specify the cache location
-- Level 1 --> a cache is associated to a core.
-- Level 2 --> a cache is associated to a processor
Cache_Level : aadlinteger applies to (memory, system);

-- Specify the size of a cache
Cache_Size : aadlinteger applies to (memory, system);

-- Specify the size of a cache line
Line_Size : size applies to (memory, system);

-- Specify associativity of a cache
-- Associativity = 0    : Fully associative cache
-- Associativity = 1    : Direct mapped cache
-- Associativity =>0    : Set associative cache
Associativity : aadlinteger applies to (memory, system);

-- Specify an upper-bound on the time it takes
-- to load a memory block into the cache
Block_Reload_Time : Time_Range applies to (memory, system);

-- Specify the replacement policy of a cache
-- Direct mapped cache (Associativity = 1)
--   does not require a replacement policy.
-- Fully associative (Associativity = 0) and set associative
--   cache (Associativity > 1) do require a replacement policy.
Supported_Replacement_Policy_Type : type enumeration (
  Random, LRU, LRR, FIFO);
Replacement_Policy_Type : Supported_Replacement_Policy_Type
  applies to (memory, system);

-- Specify management applied to a cache
Supported_Coherence_Protocol_Type : type enumeration (
  Private_Cache, Shared_Cache,
  Private_Invalid_Cache, Private_MSI,
  Private_MESI);
Coherence_Protocol_Type : Supported_Coherence_Protocol_Type
  applies to (memory, system);

-- Specify the write policy applied to a cache
Supported_Write_Policy_Type : type enumeration (
  Write_Back, No_Allocated_Write_Through,
  Allocated_Write_Through, Write_Through);
Write_Policy_Type : Supported_Write_Policy_Type
  applies to (memory, system);
```

Figure 8: Properties assigned to cache components

3.4.1 Memory Hierarchy Units

Among memory hierarchy units, cache units are the one that have raised the highest interest in schedulability analysis. One of the most investigated interference related to cache unit is the Cache Related Preemption Delay (CRPD). By definition, CRPD is the additional time to refill the cache with memory blocks evicted by a preemption. It has been shown that CRPD may represent up to 44% of the WCET [15] (Worst Case Execution Time). Such overhead can be bounded [16] and Figure 8 shows a sample of AADL properties introduced for cache analysis. Figure 9 is an example of a dual-core processing unit with L1 cache units.

```
memory cache end cache;
memory implementation cache.impl
properties
  Memory_Size=>1024Bytes;
  Line_Size=>16Bytes;
  Cache_Type=>Instruction_Cache;
  Cache_Level=>1;
  Associativity=>1;
  Cache_Size=>1024;
  Block_Reload_Time=>1 us .. 2 us;
end cache.impl;

system dual_core end dual_core;
system implementation dual_core.impl
subcomponents
  core1 : processor uni_core;
  core2 : processor uni_core;
  core1_L1_ICache : memory cache.impl;
  core2_L1_ICache : memory cache.impl;
properties
  System_Soc_Type=>SoC_Processing_Unit;
  SoC_Interconnection_Type=>Shared_Bus;
end dual_core.impl
```

Figure 9: Example of a processing unit with L1 caches

3.4.2 Interconnections

The interconnection of IPs is a major issue in the design of complex SoC (System On Chip). The basic solution using a simple shared bus is known to be non scalable to regards with the number of devices connected. Then, complex solutions, like hierarchical buses or Networks on Chip, have to be implemented to meet the communication requirements.

```
-- Specify how are connected IPs
Supported_SoC_Interconnection_Type : type enumeration (
  Shared_Bus, Star, Point_to_Point_Link , Crossbar,
  NoC, Torus_NoC, Mesh_NoC);
SoC_Interconnection_Type : Supported_SoC_Interconnection_Type
  applies to (system);

-- In case of a NoC, specify its routing protocol
Supported_Routing_Type : type enumeration (
   XY, YX, XYX, Programmable_Routing);
Routing_Protocol : Supported_Routing_Type
  applies to (system);

-- In case of a NoC, specify its switching protocol
Supported_Switching_Type : type enumeration (
  Wormhole, Store_And_Forward, Virtual_Cut_Through);
Switching_Protocol : Supported_Switching_Type
  applies to (system);
```

Figure 10: Interconnection properties

The interconnection system and the scheduling of the exchanges on this system have an impact on the overall performance and on the predictability. Figure 10 presents a sample of the properties we defined to specify the details of interconnections. COTS manufacturers can be hesitant to disclose

the detail of the interconnection architecture and its management policy to protect their intellectual property. This lake of transparency can be a problem in the context of a certification process [17].

4 Schedulability Analysis Features

After about 25 years of research, schedulability of multiprocessors stays challenging. Many approaches have been proposed [18], however no general solution exists.

In the case of multicore architectures, authors proposed the design of dedicated software/hardware architectures limiting/removing interference. Other proposed extensions of uniprocessor analysis methods with interference from shared hardware resources. For instance, the worst-case response time (WCRT) analysis has been extending with CRPD analysis when cache units are present. WCRT analysis can lead to proofs. Unfortunately, combining extensions of WCRT analysis to support multiple sources of interference is not always possible or may lead to pessimistic analysis.

We have implemented scheduling analysis methods based on scheduling simulations. Scheduling simulation usually cannot lead to schedulability proofs. The result can be considered as a schedulability proof for a given system model when the scheduling simulation is proved to be sustainable and the computed on its feasibility interval.

In [7] and [6], the authors respectively defined the concepts of feasibility interval and sustainability as follows:

Definition 1 *(Feasibility interval) A feasibility interval is defined as a finite interval such that if all the deadlines of jobs released in the interval are met, then the system is schedulable.*

Definition 2 *(Sustainability) A schedulability test for a scheduling policy is sustainable if any system deemed schedulable by the schedulability test remains schedulable when the parameters of one or more individual job(s) are changed in any, some, or all of the following ways: (i) decreased execution requirements; (ii) later arrival times; (iii) smaller jitter; and (iv) larger relative deadlines.*

To perform schedulability analysis of AADL multiprocessor models, users can use the OSATE and Cheddar tools. OSATE implements transformation rules to convert an AADL model into a Cheddar model. Once the Cheddar model is generated, users can use the following features of Cheddar:

- Compute the scheduling of the AADL threads with one of the 15 multiprocessor scheduling policies implemented in Cheddar. We implemented partitioned and global classical policies (e.g. fixed priority, EDF or LLF). specific multiprocessor policies (e.g. EDZL, Pfair or RUN), hierarchical policies (e.g. multicore ARINC653 scheduling) and mixed criticality policies (e.g. AMC).
- Feasibility intervals presented in [7] have been implemented in Cheddar and the tool automatically selects one methods compliant with the AADL model to analyze. The tool however cannot guaranty that the shortest feasibility interval is actually applied.
- Version 3.3 of Cheddar does not currently verify if the AADL model is sustainable. Sustainability is the responsibility of the users.
- Finally, scheduling simulations can be combined with interference measurements or analysis. Cheddar 3.3 provides cache L1 interference models [19], a DRAM model based on [20], Kalray MPPA-256 memory model [21], and a NoC model [22].

5 Related work

The modeling of multiprocessor architectures for schedulability analysis has received much attention in the last two decades from the AUTOSAR, OMG (with MARTES) and SAE (with AADL) communities.

In the AUTOSAR community, the work in [23] presents AUTOSAR version 4.0, which supports partitioned multicore systems. Intercore communications are designed with spinlocks, which have to be taken into account in the schedulability analysis. In [24], the authors investigate starvations and deadlocks when synchronizations are achieved with spinlocks. The study in [25] proposes optimization methods to reduce synchronization between cores. A last example of multicore support with AUTOSAR is the work presented in [26], which designed a method to generate off-line scheduling for dependent periodic task sets.

In the OMG community, schedulability of multiprocessor architectures has been investigated with the MARTE UML profile. MARTE enables modeling and analysis from single-core up to partitioned multicore architectures. In [27], the authors extend MARTE stereotypes to enable task migrations and therefore the analysis of multicore systems with global scheduling. The work in [28] shows how to translate a model from MARTE to SimSo [29], a schedulability tool allowing the analysis of multiprocessor architectures with global or semi-partitioned scheduling. Finally, in [30], the authors extend the MARTE profile by defining hardware architecture concepts so that performances can be investigated by SystemC simulations.

We focus on the AADL standard, on which several works have also been conducted on the modeling and the analysis of multiprocessor architectures [31, 32, 33, 34, 35, 36, 37].

In [31], the authors propose a AADL standard annex to model multiprocessor architectures compliant with the ARINC 653 standard. They describe means to model TSP (Time and Space Partitioning) architectures and to evaluate their schedulability by simulations. TSP is also investigated in [34] where extensions of the TASTE toolset, an AADL backed tool promoted by the ESA (European Space Agency), have been proposed for modeling/analysis of SMP multicore platforms.

While AADL examples above focused on space and avionics systems, multiprocessor architectures are not limited to these domains. For example, the work in [35] investigates the AADL engineering of railway systems that may embed single-core or multicore platforms. ROS robotic applications may also be run on multiprocessor platforms, and the authors of [36] proposed models of them in order to predict processor usage.

Multiprocessor SoC can be designed by FPGA configurations. The study in [37] presents AADL extensions for design space exploration of FPGA configurations.

Finally, proposals above are related to standards (UML, AADL or AUTOSAR), but others have designed specific languages or models for such a purpose. As an example, in [33], the authors exploit a modeling language called PML (PHYLOG Modeling Language) to model multicore platforms for certification activities.

6 Conclusion

In this article, we addressed multiprocessor architectures schedulability with the AADL language. We have proposed AADL guidelines to model such architectures. Schedulability of multiprocessor stays challenging. We propose to use simulation-based analysis of these models that may produce schedulability proofs when sustainability is enforced and when simulations are run on the feasibility interval.

To experiment the previous proposals, we have implemented an OSATE plugin to achieve model transformation from AADL to CheddarADL [38], allowing users to perform schedulability analysis with Cheddar. These modeling and verification proposals have been used for the PDR (Preliminary Design Review) of the ESA PLATO satellite [39].

For future work, we plan to focus on shared hardware resources on space-hardened multiprocessor boards, which may lead to adaptations of the AADL modeling guidelines. Proposals presented in this article will be evaluated on GR740 boards, using the POMELOS platform (http://beru.univ-brest.fr/pomelos).

Acknowledgment

This work and Cheddar are carried out with the support of Ellidiss Technologies, University of Brest, Lab-STICC UMR CNRS 6285, Conseil régional de Bretagne, Brest Métropole, BPI France, Conseil département du Finistère, Thales DMS/GIS Cormorant, CNES, EGIDE/Campus France PESSOA number 27380SA and 37932TF, MOSAR Horizon 2020 project under grant agreement No 821996.

References

[1] F. J. Pollack, "New microarchitecture challenges in the coming generations of CMOS process technologies (keynote address)," in *Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture*, p. 2, IEEE Computer Society, 1999.

[2] J. Swingler and J. McBride, "The synergistic relationship of stresses in the automotive connector," in *Proceedings of the 19th International Conference on Electric Contact Phenomena*, pp. 141–145, 1998.

[3] B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis, "A comprehensive survey of industry practice in real-time systems," *Real-Time Systems*, vol. 58, no. 3, pp. 358–398, 2022.

[4] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The architecture analysis & design language (aadl): An introduction," tech. rep., 2006.

[5] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," *ACM SIGAda Ada Letters*, vol. 24, no. 4, pp. 1–8, 2004.

[6] S. Baruah and A. Burns, "Sustainable scheduling analysis," in *2006 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, pp. 159–168, IEEE, 2006.

[7] J. Goossens, E. Grolleau, and L. Cucu-Grosjean, "Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms," *Real-time systems*, vol. 52, no. 6, pp. 808–832, 2016.

[8] P. Feiler, "The open source AADL tool environment (osate)," 2019.

[9] K. Basu, A. Choudhary, J. Pisharath, and M. Kandemir, "Power protocol: reducing power dissipation on off-chip data buses," in *Proceedings of the 35th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 345–355, June 2002.

[10] S. Ma, L. Huang, M. Lai, and W. Shi, *Networks-on-Chip: From implementation to programming paradigms*. Morgan Kaufmann, 2014.

[11] T. Ungerer, B. Robič, and J. Šilc, "A survey of processors with explicit multithreading," *ACM Computing Surveys (CSUR)*, vol. 35, no. 1, pp. 29–63, 2003.

[12] P. Huyck, "ARINC 653 and multi-core microprocessors—considerations and potential impacts," in *Proceedings of the IEEE/AIAA* 31^{st} *Digital Avionics Systems Conference (DASC)*, pp. 1–16, Oct. 2012.

[13] R. Fuchsen, "How to address certification for multi-core based IMA platforms: Current status and potential solutions," *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*, pp. 5.E.3–1, Oct. 2010.

[14] S. Rubini, V.-A. Nicolas, F. Singhoff, A. Plantec, H. N. Tran, and P. Dissaux, "Specification of schedulability assumptions to leverage multiprocessor analysis," *Journal of Systems Architecture*, vol. 133, p. 102761, 2022.

[15] R. Pellizzoni and M. Caccamo, "Toward the predictable integration of real-time cots based systems," in *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pp. 73–82, IEEE, 2007.

[16] M. Lv, N. Guan, J. Reineke, R. Wilhelm, and W. Yi, "A survey on static cache analysis for real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 3, no. 1, pp. 05–1, 2016.

[17] L. M. Kinnan, "Use of multicore processors in avionics systems and its potential impact on implementation and certification," in *2009 IEEE/AIAA 28th Digital Avionics Systems Conference*, pp. 1–E, IEEE, 2009.

[18] C. Maiza, H. Rihani, J. M. Rivas, J. Goossens, S. Altmeyer, and R. I. Davis, "A survey of timing verification techniques for multi-core real-time systems," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–38, 2019.

[19] H. N. Tran, S. Rubini, J. Boukhobza, and F. Singhoff, "Feasibility interval and sustainable scheduling simulation with CRPD on uniprocessor platform," *Journal of Systems Architecture*, vol. 115, p. 102007, 2021.

[20] H. Kim, D. De Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar, "Bounding memory interference delay in COTS-based multi-core systems," in *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 145–154, IEEE, 2014.

[21] H. N. Tran, A. Honorat, J.-P. Talpin, T. Gautier, and L. Besnard, "Efficient contention-aware scheduling of sdf graphs on shared multi-bank memory," in *2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pp. 114–123, IEEE, 2019.

[22] M. Dridi, F. Singhoff, S. Rubini, and J.-P. Diguet, "ECTM: A network-on-chip communication model to combine task and message schedulability analysis," *Journal of Systems Architecture*, vol. 114, p. 101931, 2021.

[23] N. Böhm, D. Lohmann, and W. Schröder-Preikschat, "Multi-core processors in the automotive domain: An autosar case study," *Proceedings Work-in-Progress Session*, pp. 1–6, 2010.

[24] K. S. Lakshmanan, G. Bhatia, and R. Rajkumar, "Autosar extensions for predictable task synchronization in multi-core ecus," tech. rep., SAE Technical Paper, 2011.

[25] G. Entrup, A. Kässens, B. Fiedler, and D. Lohmann, "Applied static analysis and specialization of cross-core syscalls for multi-core autosar os," *Real-Time Systems*, vol. 60, no. 3, pp. 491–533, 2024.

[26] W. Wang, F. Camut, and B. Miramond, "Generation of schedule tables on multi-core systems for autosar applications," in *2016 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pp. 191–198, IEEE, 2016.

[27] A. Magdich, Y. H. Kacem, A. Mahfoudhi, and M. Abid, "A marte extension for global scheduling analysis of multiprocessor systems," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering*, pp. 371–379, IEEE, 2012.

[28] A. Magdich, Y. H. Kacem, A. Mahfoudhi, and M. Abid, "From uml/marte models of multiprocessor real-time embedded systems to early schedulability analysis based on simso tool.," in *ICSOFT-EA*, pp. 202–209, 2016.

[29] M. Chéramy, P.-E. Hladik, and A.-M. Déplanche, "Simso: A simulation tool to evaluate real-time multiprocessor scheduling algorithms," in *5th international workshop on analysis tools and methodologies for embedded and real-time systems (WATERS)*, pp. 6–p, 2014.

[30] É. Piel, R. B. Atitallah, P. Marquet, S. Meftali, S. Niar, A. Etien, J.-L. Dekeyser, P. Boulet, and I. Europe, "Gaspard2: from marte to systemc simulation," in *proc. of the DATE*, vol. 8, 2008.

[31] J. Delange, L. Pautet, A. Plantec, M. Kerboeuf, F. Singhoff, and F. Kordon, "Validate, simulate, and implement ARINC653 systems using the AADL," in *Proceedings of the ACM SIGAda annual international conference on Ada and related technologies*, pp. 31–44, 2009.

[32] T. Waegemann, T. Langer, J. Mottok, L. Osinski, F. Stappert, and R. T. Kolagari, "Models for dependable heterogenous multi-and many-core system software design revisited," in *ARCS 2016; 29th International Conference on Architecture of Computing Systems*, pp. 1–8, VDE, 2016.

[33] F. Boniol, J. Brunel, K. Delmas, C. Pagetti, and V. Jegu, "Modelling and analyzing multi-core COTS processors," in *11th European Congress on Embedded Real Time Software and Systems (ERTS 2022)*, 2022.

[34] C. Honvault, J. Hugues, and C. Pagetti, "Model-based design, analysis and synthesis for TSP multi-core space systems," in *ERTS 2018*, 2018.

[35] P. Crisafulli, D. Blouin, F. Caron, and C. Maxim, "Engineering railway systems with an architecture-centric process supported by AADL and ALISA: an experience report," in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, 2020.

[36] S. Eric and L. Bourdon, "Introducing cpu load analysis from AADL models for ros applications: a use case," in *2021 Forum on specification & Design Languages (FDL)*, pp. 01–08, IEEE, 2021.

[37] D. Blouin, D. Chillet, E. Senn, S. Bilavarn, R. Bonamy, and C. Samoyeau, "AADL extension to model classical FPGA and FPGA embedded within a soc," *International Journal of Reconfigurable Computing*, vol. 2011, no. 1, p. 425401, 2011.

[38] C. Fotsing, F. Singhoff, A. Plantec, V. Gaudel, S. Rubini, S. Li, H. N. Tran, L. Lemarchand, P. Dissaux, and J. Legrand, "Cheddar architecture description language," *Lab-STICC technical report*, 2014.

[39] P. Plasson, G. Brusq, F. Singhoff, H. N. Tran, S. Rubini, and P. Dissaux, "Plato n-dpu on-board software: an ideal candidate for multicore scheduling analysis," in *11th European Congress ERTSS Embedded Real Time Software and System*, 2022.

Annex

Property sets defined for the modeling of multiprocessor architectures can be accessed on:

```
http://beru.univ-brest.fr/svn/CHEDDAR/trunk/
src/osate/osate2-cheddar_properties_plugin/
resources/properties/Cheddar/
```

Two property sets are related to multiprocessor schedulability analysis. Property set `Cheddar_Properties` defines AADL properties that are common for schedulability analysis of any kinds of processor architectures. Property set `Cheddar_Multicore_Properties` contains AADL properties specifically designed for multiprocessor architectures.

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