The journal for the international Ada community

Ada User Journal

Volume 42
Number 2
June 2021

Produced by Ada-Europe

Information on Subscriptions and Advertisements

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

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

ADA USER JOURNAL

Volume 42
Number 2
June 2021

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

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

Original Papers

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

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

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

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

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

Proceedings and Special Issues

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

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

News and Product Announcements

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

Reprinted Articles

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

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

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

Editorial

With this issue we bring what we believe are good news for most of our subscribers concerning the distribution of the physical copies of the AUJ. Copies will now be sent directly to each subscriber from the printer, instead of being sent firstly, in bulk, to each national body and then, from there, to each destination. For direct members nothing will change. We believe that this simplification is beneficial for everyone.

Concerning this June issue, we start the publication of the articles presented at Work-In-Progress (WiP) Session of the Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021). This year the conference took place as a virtual event, which was a new and quite positive experience for many attendants. Hopefully it will be possible to regain the physical presence next year, but even if not, we can now rest assured that running the conference virtually is a viable option.

The six articles included in the issue are the following. Firstly, a contribution authored by A. Amurrio and E. Azketa from the Ikerlan Technology Research Centre, and by M. Aldea Rivas and J. Javier Gutiérrez, from the University of Cantabria. The article investigates the influence of partition windows on the task response time in hierarchically scheduled time partitioned distributed systems. Then, the article "Adoption of ACPS in Nuclear Reactor Analysis", authored by C. Castagna, D. Cancila and A. Cammi, looks at the problem of bounding the uncertainty in the design of these safety-critical CPS, which is made harder due to its multidisciplinary nature. The third article is single authored, by James S. Kimmet from Raytheon Missiles & Defense, describing a data store that was developed to be used in concurrent real-time systems, ensuring that the shared data is kept always coherent. The fourth article provides another contribution coming from the University of Cantabria, in this case authored by M. Aldea Rivas and H. Pérez Tijero. The article provides the status of an ongoing project that aims at porting M2OS, a small the real-time operating system, to the Arduino Uno platform. Then, the article "Fuzion – Safety through Simplicity", by Dr. Fridtjof Siebert from Tokiwa Software GmbH, introduces the Fuzion programming language, which provides several interesting features. The language combines a powerful syntax with safety features, meant for high-performance safety-critical applications. Finally, yet another paper from the University of Cantabria, by H. Pérez Tijero, D. García Prieto, and J. Javier Gutiérrez, that introduces a new solution for exploiting EDF scheduling and synchronized clocks in a distributed real-time system, to consider end-to-end global deadlines and achieve improvements in resource utilization.

As usual, the issue includes the Quarterly News Digest, prepared by Alejandro R. Mosteo, and the Calendar and Events sections, prepared by Dirk Craeynest. The issue closes with another and final note by John Barnes on puzzles that have been presented in past editions of the Ada-Europe conference. The solution for the pyramids puzzle from the March issue is provided, and the four puzzles mentioned at this year's conference, along with their solutions, are also included in the note. We are very grateful to John Barnes for his continued contribution to the AUJ over the past year. Thank you!

Antonio Casimiro
Lisboa
June 2021
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

[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,

This period saw the celebration of the Ada-Europe conference, after a year of hiatus, and only in virtual form. This is cause for celebration and a signal of hope for Adaists to meet again in the future in this close-knit event where one can mingle with newcomers, old faces, and the "guiding lights" of the language alike. Announcements about the event, for the record, are found in this number [1].

Another yearly moment of excitement for the Ada open source community is the release of the GNAT Community Edition, which we witnessed at the end of May [2]. And, speaking of open source communities, I will mention the mass exodus from the Freenode chat network due to a change in ownership and policies. The dwellers of #ada have chosen, on the 20th anniversary of the channel, the Libera Chat network as a new home. The thread announcing the news [3] is also a reminiscence about other venerable technologies that, like IRC and Usenet, still are going around.

In another curious development, a mostly off-topic thread that had seen its last post in 2014 was somehow revived, and I cannot resist reading about the computing anecdotes of times long past, in this case framed in a "Pascal vs C" context [4].

The polemic topic about Ada itself in this number was Unicode, or Ada lackluster support thereof, according to some. Opinions, hopes for a brighter future, and insights on how it came to be in its present form are discussed in [5].

Sincerely,
Alejandro R. Mosteo.

[1] "Ada-Europe Int. Conf. on Reliable Software Technologies, AEiC 2021", in Ada-related Events.

[2] "GNAT CE 2021", in Ada-related Tools.

[3] "Ada IRC Channel Migrates to Libera Chat", in Ada Practice.

[4] "Pascal vs C Language Families", in Ada and Other Languages.

[5] "Ada and Unicode", in Ada Practice.

Ada-related Events

Ada-Europe Int. Conf. on Reliable Software Technologies, AEiC 2021

[Past event, for the record. —arm]

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe Int.Conf. Reliable Software Technologies, AEiC 2021
Date: Tue, 27 Apr 2021 17:46:30 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

Call for Participation

*** PROGRAM SUMMARY ***

25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021)

7-10 June 2021, Virtual Event

www.ada-europe.org/conference2021

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

#AEiC2021 #AdaEurope #AdaProgramming

General Information

The 25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021), initially scheduled to take place in Santander, Spain, will be held online from the 7th to the 10th of June 2021, using the underline.io conference platform.

The conference program includes parallel tutorials on Monday 7th, and a technical program and vendor exhibition from Tuesday to Thursday. The conference also includes breaks and virtual social events that will allow networking among the participants.

Overview of the Week

Monday 7th

- Welcome Social Event
- 5 Parallel Tutorials
- Ice-Breaking Social Event

Tuesday 8th

- Ice-Breaking Social Event and Opening
- Techn. Session 1: Scheduling and mixed-criticality systems
- Keynote 1
- Techn. Session 2: Software modeling
- Social Event

Wednesday 9th

- Welcome Social Event
- Techn. Session 3: Autonomous systems
- Work-in-Progress Session
- Keynote 2
- Techn. Session 4: Ada issues and Ravenscar
- Social Event

Thursday 10th

- Welcome Social Event
- Techn. Session 5: Validation and verification tools
- Techn. Session 6: Emerging applications with reliability requirements
- Keynote 3
- Techn. Session 7: Safety challenges
- Best Presentation Award, Closing Session and Party

The program runs between 12:30 and 18:30 CEST, to allow participation from different time zones. For full details and

up-to-date information, see the conference web page: http://www.ada-europe.org/conference2021

Keynote Talks

In each of the three main conference days, a keynote will be delivered to address hot topics of relevance in the conference scope, with ample time for questions and answers. The keynotes will be:

- Ángel Conde, Data Analytics and Artificial Intelligence team leader at IKERLAN (Spain), who will present his work on "Software reliability in the Big Data era with an industry-minded focus".
- Alfons Crespo, who is with the Institute of Automation and Industrial Informatics of the Universitat Politècnica de València (Spain), will give an answer to the question "Why hypervisor-based approach is the best alternative for mixed-criticality systems".
- Tucker Taft, who is Director of Language Research at AdaCore (USA), will talk on "A sampling of Ada 2022".

Technical Sessions

Given the current sanitary situation and the need to resort to a virtual format for the conference, we will all experience the advantages and benefits of exploring new formats. The technical sessions are designed with the flipped-conference concept, where the audience can access the pre-recorded presentation materials in advance and the live sessions are devoted to short presentations of the highlights of each contribution, allowing ample time for questions and answers with the presenter. The recorded materials will also be available for some time after their sessions. The technical sessions include papers submitted to the journal track that are heading towards final acceptance and open-access publication, together with industrial, invited and vendor presentations.

Work-in-Progress Session

The Work-in-Progress session contains contributions of evolving and early-stage ideas, or new research directions. They are presented in a special session consisting of a round of very short presentations of the highlights of each contribution, followed by a poster session in the same virtual space where the breaks are held.

Exhibition

From Tuesday to Thursday the conference platform will provide access to virtual booths where participants will be able to find information on the conference exhibitors and chat with them or request meetings. The virtual break lounge where the breaks and social events will take place will also have a space for meeting with the exhibitors.

Tutorials

Five four-hour parallel tutorials are offered on Monday 7th:

- TU-1: Programming mobile robots with ROS2 and the RCLAda Ada client library, by Alejandro R. Mosteo
- TU-2: Introduction to the development of safety critical software, by Jean-Pierre Rosen
- TU-3: Parallel programming with Ada and OpenMP, by Sara Royuela, S. Tucker Taft, Luis Miguel Pinho
- TU-4: Timing verification from UML & MARTE design models: techniques & tools, by Laurent Rioux, Julio Medina and Shuai Li
- TU-5: Programming shared memory computers, by Jan Verschelde

Social Program

The virtual conference platform will offer a space under the gather.town environment to allow informal and lively gathering of the participants. This space may have different areas, such as rooms, tables, and corners where a participant can approach to talk through videoconferencing with participants in the same virtual area. This facility will be used for the breaks, poster session, exhibition and social events. Particular themes for some of the social events will be announced in the conference platform and in the web page.

Further Information

Participation for the full event, including tutorials, is free for Ada-Europe members and only 60 EUR for all others. Registration is required for all. The conference web page will shortly give full and up-to-date details on the program, the registration process and the virtual platform: http://www.ada-europe.org/conference2021

AEiC 2021 Sponsors

- AdaCore: https://www.adacore.com/
- Ellidiss: https://www.ellidiss.com/
- PTC: http://www.ptc.com/developer-tools
- Universidad de Cantabria: https://web.unican.es/en/
- Vector: https://www.vector.com/at/en/

The conference is supported and sponsored by

- Ada-Europe: http://www.ada-europe.org/

and organized in cooperation with

- ACM SIGAda: http://www.sigada.org/
- ACM SIGBED: https://sigbed.org/
- ACM SIGPLAN: http://www.sigplan.org/
- ARA: https://www.adaic.org/community/

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2021 Publicity Chair (aka Ada-Europe 2021)

Dirk.Craeynest@cs.kuleuven.be

(V4.1)

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Fri, 30 Apr 2021 07:59:16 -0000

Registration site for AEiC2021 is online: registration.ada-europe.org.

Register now for the 25th Ada-Europe Int'l Conference on Reliable Software Technologies!

Press Release - AEiC 2021, Ada-Europe Reliable Softw. Technol.

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Press Release - AEiC 2021, Ada-Europe Reliable Softw. Technol.
Date: Sun, 30 May 2021 16:22:28 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada,comp.lang.misc

FINAL Call for Participation

*** UPDATED Program Summary ***

25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021)

7-10 June 2021, Virtual Event

www.ada-europe.org/conference2021

*** Check out tutorials! ***

www.ada-europe.org/conference2021/tutorials.html

*** Don't miss the thematic social events on Tuesday and Wednesday ***

*** Full Program available on the conference web site ***

*** Register now! ***

#AEiC2021 #AdaEurope #AdaProgramming

Press release:

25th Ada-Europe Int'l Conference on Reliable Software Technologies
International experts meet in virtual conference hosted by Underline
Santander, Spain (31 May 2021) - Ada-

Europe together with the University of Cantabria, Spain organize from 7 to 10 June 2021 the 25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2021). The conference was initially scheduled to take place in Santander, Spain. According to the safety and sanitary measures under the COVID-19 pandemic, this year the conference will be a virtual event, hosted by Underline (https://underline.io). The event is in cooperation with the Ada Resource Association (ARA), and with ACM's Special Interest Groups on Ada (SIGAda), on Embedded Systems (SIGBED) and on Programming Languages (SIGPLAN).

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

This year's conference offers 5 tutorials, 3 keynotes, a technical program of 7 sessions with refereed papers, invited and industrial presentations, a work-in-progress session, an industrial exhibition and vendor presentations, and a social program.

Five parallel tutorials are scheduled on Monday, targeting different audiences:

- "Programming mobile robots with ROS2 and the RCLAda Ada client library", by Alejandro R. Mosteo;
- "Introduction to the development of safety critical software", by Jean-Pierre Rosen;
- "Parallel programming with Ada and OpenMP", by Sara Royuela, S. Tucker Taft, Luis Miguel Pinho;
- "Timing verification from UML & MARTE design models: techniques & tools", by Laurent Rioux, Julio Medina and Shuai Li;
- "Programming shared memory computers", by Jan Verschelde.

Tutorial registration is complementary for conference participants.

The industrial exhibition opens Tuesday under the Expo area in the virtual platform and also in the Lounge, which is the networking area. It runs until the end of Thursday afternoon. Exhibitors include AdaCore, PTC Developer Tools, and Ada-Europe. All conference participants are invited to the exhibition as well as to the virtual social events.

Three eminent speakers have been invited to deliver a keynote at each of the core conference days:

- Ángel Conde, Data Analytics and Artificial Intelligence team leader at IKERLAN (Spain), who will present his work on "Software reliability in the Big Data era with an industry-minded focus";
- Alfons Crespo, who is with the Institute of Automation and Industrial Informatics of the Universitat Politècnica de València (Spain), will give an answer to the question "Why hypervisor-based approach is the best alternative for mixed-criticality systems";
- Tucker Taft, who is Director of Language Research at AdaCore (USA), will talk on "A sampling of Ada 2022".

The technical program from Tuesday to Thursday presents 13 refereed technical papers and 5 invited, 6 industrial and 4 vendor presentations in sessions on:

- Scheduling and mixed-criticality systems,
- Software modeling,
- Autonomous systems,
- Ada issues and Ravenscar,
- Validation and verification tools,
- Emerging applications with reliability requirements,
- Safety challenges.

In addition, there is a work-in-progress session including 8 presentations and associated posters.

Peer-reviewed papers have been submitted to a special issue of the Journal of Systems Architecture and are heading towards final acceptance as open-access publications. Industrial and work-in-progress presentations, together with tutorial abstracts, will be offered publication in the Ada User Journal, the quarterly magazine of Ada-Europe.

The social program is hosted in a space under the gather.town environment that allows informal and lively gathering of the participants. This space has different areas, such as rooms, tables, and corners where a participant can approach to talk through videoconferencing with participants in the same virtual area. This facility will be used for the breaks, poster session, exhibition and social events. Don't miss the thematic social events at the end of each core conference day.

The Best Presentation Award will be offered during the Closing session.

The full program is available on the conference web site. Online registration is still possible.

Latest updates:

The "Final Program" is available at www.ada-europe.org/conference2021 /final-program.pdf.

Check out the tutorials in the PDF program, or in the schedule at

www.ada-europe.org/conference2021/ tutorials.html.

Registration fees are lower than ever and the registration process is done on-line. Don't delay for all details, select "Registration" at www.ada-europe.org/ conference2021 or go directly to https://registration.ada-europe.org.

The technical sessions are designed with the flipped-conference concept, where the audience can access pre-recorded presentation materials in advance. The live sessions are devoted to short presentations of the highlights of each contribution, allowing ample time for questions and answers with the presenter. The recorded materials will also be available for some time after their sessions.

The program runs between 12:30 and 18:30 CEST, to allow participation from different time zones. For more info and latest updates see the conference web site at www.ada-europe.org/conference2021.

AEiC 2021 is sponsored by AdaCore (www.adacore.com), Ellidiss (www.ellidiss.com), PTC Developer Tools (www.ptc.com/developer-tools), Universidad de Cantabria (web.unican.es/en), and Vector (www.vector.com/at/en).

Help promote the conference by advertising it.

Recommended Twitter hashtags: #AdaEurope and/or #AEiC2021.

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2021 Publicity Chair (aka Ada-Europe 2021),

Dirk.Craeynest@cs.kuleuven.be

* 25th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2021)

* June 7-10, 2021 * online event * www.ada-europe.org/conference2021 **

(V6.1)

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Sun, 6 Jun 2021 10:11:09 -0000

If you plan to attend one of the #AEiC2021 #tutorials on Mon 7 Jun, don't forget to check the prerequisites: you may have to download material preferably before the tutorial starts.

See you at #AdaEurope's #OnlineConference soon!

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Thu, 10 Jun 2021 07:43:05 -0000

#AEiC2021 #AdaEurope #OnlineConference #AdaProgramming

Don't miss today's keynote!

Tucker Taft will present "A sampling of Ada 2022"

Abstract:

The forthcoming Ada 2022 revision of the Ada standard includes significant new features, which together make the language more expressive and productive in a multicore context, while enhancing its safety and support for more complete abstractions with formal contracts.

This talk will introduce these key new features with a series of examples:

- parallel loops and blocks, coupled with static detection of data races and potential blocking
- iterator syntax for incorporating filters and user-defined iterator procedures-libraries for atomic operations, including fetch-and-add and compare-and-swap
- aggregates, literals, images, and map-reduce for user-defined types
- libraries for arbitrary precision integer and rational arithmetic
- more expressive contracts using delta aggregates and declare expressions
- the Jorvik profile as the next step up from the Ravenscar profile

Call for Paper in CodeLand

From: Mockturtle <framefritti@gmail.com>
Subject: Call for Paper in CodeLand
Date: Sat, 26 Jun 2021 09:04:37 -0700
Newsgroups: comp.lang.ada

I just "tripped over" a CFP for the event CodeLand (link at the end). They accept proposals for 15-minutes pre-recorded video talk; deadline 20th of July.

CodeLand's primary audience is early-career programmers and their mentors. Among the themes I see "Technical Deep Dives" and "Path to Programmer" that could maybe be suitable for something Ada-related.

Is maybe someone interested in proposing something? I am going to think about it...

Summary of the most important info

* When: September 23-24, 2021
* How: virtual only, pre-recorded video
* Deadline: July 20, 2021 at 11:59pm UTC.
* Acceptance/reject date: August 17, 2021.
* Themes
  - Code for Good
  - Early-career confidence
  - Open source strong
  - Path to programmer
  - Technical deep dives
* More info: https://cfp.codelandconf.com/events/codeland-2021

Accepted speakers will be asked to participate in a panel session (suggested but not required). They should be prepared to answer moderated attendee questions about their talk.

New Competition: Ada/SPARK Crate of the Year Award

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Subject: New competition: Ada/SPARK Crate Of The Year Award
Date: Mon, 28 Jun 2021 03:11:38 -0700
Newsgroups: comp.lang.ada

I am happy to announce AdaCore's new programming competition: The Ada/SPARK Crate Of The Year Award!

The announcement is here: https://blog.adacore.com/announcing-the-first-ada-spark-crate-of-the-year-award

And you can register here: https://github.com/AdaCore/Ada-SPARK-Crate-Of-The-Year

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Wed, 30 Jun 2021 04:44:18 -0700

A candidate project must be on github? (fair enough)

Any Alire crate project must be on github?

Thanks.

From: Fabien Chouteau <fabien.chouteau@gmail.com>
Date: Wed, 30 Jun 2021 05:47:49 -0700

For the Alire community index, your crate can either be in tarball format and hosted anywhere you want, or it can be a commit in a git repo in which case we ask you to host it on GitHub, SourceForge, GitLab or Bitbucket.

Ada Semantic Interface Specification (ASIS)

ASIS and libadalang

From: J-P. Rosen <rosen@adalog.fr>
Subject: ASIS for Gnat (was: Any chance of programming a web frontend in Ada 2012?
Date: Wed, 9 Jun 2021 07:02:40 +0200
Newsgroups: comp.lang.ada

> I did some progress in this direction, but ASIS4GNAT is abandoned and my project is suspended.

ASIS4GNAT is not abandoned, it is just not part of the CE edition. Pro users have access to it.

Please drop me a note if you have developed an ASIS tool, or are using an ASIS-based tool. With enough protests, we may convince AdaCore to make ASIS4GNAT available to the community.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 11 Jun 2021 11:47:12 -0700

> ASIS4GNAT is not abandoned, it is just not part of the CE edition. Pro users have access to it.

On the other hand, if you are starting a new project, libadalang is a better choice.

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 11 Jun 2021 22:31:25 +0200

> On the other hand, if you are starting a new project, libadalang is a better choice.

What makes you think so?

By all means, compare the specifications of an ASIS package (Asis.Statements, Asis.Declarations) to Libadalang.Analysis and see which one is more usable...

From: Rod Kay <rodakay5@gmail.com>
Date: Sat, 12 Jun 2021 20:47:29 +1000

> Please drop me a note if you have developed an ASIS tool [...]

I switched from ASIS to libadalang for an Ada IDE project also, since I thought ASIS was abandoned.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 12 Jun 2021 16:04:19 +0200

If you want to analyze code while it is being typed, as is common in an IDE, Libadalang is certainly the way to go.

If you want to make sophisticated analysis tools, it's another story. Hopefully, my paper at AE will soon be available...

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 14 Jun 2021 05:37:57 -0700

Oh, I am looking forward to reading it.

BTW, I really liked your "Memory Management in Ada 2012" video; I've used it as a reference several times to explain to Rust-people that Ada is safer than expected because pointers aren't required for a lot of things, and so you don't have to worry about null-exclusion.

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Tue, 15 Jun 2021 14:40:41 -0700

Is JGNAT reliable, updated, available?

Thanks.

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 16 Jun 2021 10:41:31 +0200

> Is JGNAT

>reliable,

I didn' try it enough to answer this

> updated,

No. The latest version is 2013. Another one of the useful stuff abandoned by AdaCore.

> available?

Yes, from Adacore's community download page.

Ada and Education

Exercism.io Needs Ada

From: Bruce Axtens <bruce.axtens@gmail.com>
Subject: Exercism.io needs Ada
Date: Sat, 19 Jun 2021 13:30:37 +0800
Newsgroups: comp.lang.ada

In case anyone is looking to encourage people to learn and use Ada, Exercism.io is a good place to learn. Lots of languages already but Ada isn't one of them. Become a maintainer or a mentor.

Maintainer: https://exercism.io/become-a-maintainer

Mentor: https://exercism.io/become-a-mentor#more-info

Bruce Axtens, vbnet maintainer and students of about 26 languages none of which are, sadly, Ada.

From: Andreas Zeurcher <zuercher_andreas@outlook.com>
Date: Mon, 28 Jun 2021 13:42:03 -0700

Since you are a maintainer of another language (Visual Basic .Net), what is the precise sequence of steps that one would need to perform to become the 1st maintainer of a new presence of Ada in Exercism.io? I think that the 1st maintainer is specifically the person who bootstraps up a new language's presence on exercism.io, correct? Apparently, the sequence of steps might begin:

1) Practice being a newbie student of some arbitrary existing(-on-exercism.io) language's exercises to get the feel for how exercism.io is supposed to operate.

then

2) Contact another language's maintainer to be the 1st maintainer of Ada's fresh presence on exercism.io.

Are there more steps than that? For example, must the 1st maintainer recruit a separate 1st mentor or must the 1st maintainer act also as 1st mentor? Are those 2 steps perfectly stated or are they botched somehow? Does some sort of vetting occur for the quality of a maintainer at time of volunteering that could cause the 1st maintainer to be rejected? Does some sort of vetting for the desirability of a programming language occur up at exercism.io "corporate" that could cause Ada itself (independent of the 1st maintainer) to be rejected? It seems that Ada and (AdaSubset-with-) SPARK would be 2 separate languages on exercism.io (otherwise it gets confusing), correct? Why doesn't exercism.io have an overt webpage that answers these questions for how to bootstrap up a language's new presence on exercism.io (as this seems to be a separate & distinct topic to becoming the 2nd-or-subsequent-mentor that does have its own webpage explanation already)?

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Tue, 29 Jun 2021 06:06:56 -0700

Should not the first step be an evaluation of this site by a programming master? Maybe this has been done; if so, please inform.

Ada-related Resources

[Delta counts are from Apr 26th to Jul 22th. —arm]

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: Wed, 22 Jul 2021 11:13:21 +0100
To: Ada User Journal readership

Ada groups on various social media:

- LinkedIn: 3_161 (+42) members [1]
- Reddit: 7_104 (+678[1]) members [2]
- Stack Overflow: 2_087 (+39) questions [3]
- Libera.Chat: 76 (new[2]) users [4]
- Freenode: 15 (-79[2]) users [5]
- Gitter: 86 (+11) people [6]
- Telegram: 128 (+7) users [7]
- Twitter: 75 (+32) tweeters [8]
  74 (=) unique tweets [8]

[1] Probably caused in part by confusion with the ADA cryptocurrency.

[2] Freenode has suffered a mass exodus due to a change in ownership. Most channels have migrated to Libera.Chat.

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

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

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

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

[5] https://netsplit.de/channels/details.php?room=%23ada&net=freenode

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

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

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

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: Wed, 22 Jul 2021 11:13:21 +0100
To: Ada User Journal readership

Rosetta Code: 827 (+16) examples [1]
38 (=) developers [2]

GitHub: 763[1] (=) developers [3]

Sourceforge: 275 (+2) projects [4]

Open Hub: 214 (=) projects [5]

Alire: 171 (+15) crates [6]

Bitbucket: 89 (=) repositories [7]

Codelabs: 52 (=) repositories [8]

AdaForge: 8 (=) repositories [9]

[1] This number is unreliable due to GitHub search limitations.

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

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

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

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

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

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

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

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

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

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: Wed, 22 Jul 2021 11:13:21 +0100
To: Ada User Journal readership

[Positive ranking changes mean to go up in the ranking. The IEEE ranking deltas are in regard to the 2019 edition, as it is updated annually. —arm]

- TIOBE Index: 28 (+2) 0.48% (-0.01%) [1]

- PYPL Index: 18 (-1) 0.75% (-0.06%) [2]

- IEEE Spectrum (general): 39 (+4) Score: 32.8 (+8.0) [3]

- IEEE Spectrum (embedded): 12 (+1) Score: 32.8 (+8.0) [3]

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

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

[3] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020

Ada Domain Names

From: Heziode <heziode@protonmail.com>
Subject: Ada domain names: who holds what, and for which purpose?
Date: Thu, 24 Jun 2021 20:03:39 +0200
Newsgroups: comp.lang.ada

By curiosity, I have checked if there is some domain name related to Ada language. I mean, domains that can be officially used to represent the language (even though according to Wikipedia, it is adaic.org. Keep in mind, I am doing this out of curiosity).

I was being surprised to discover that several domains, that could be used as an "official" website (like *lang.org, *-lang.org, *.codes, where "*" is the language name), is bought but not used, and not in sales.

Here is a table of my few research:

Domain: Owner: Registered: Country: State

ada-lang.com : AdaCore : 2016/09/20 : FR*

adalang.com: ? : 2014/03/30 : US : Florida

ada-lang.org : ? : 2020/02/11 : US : WA

adalang.org : ? : 2020/02/12 : US : WA

ada.codes : Steve Arnold : 2014/04/23 : US : CA

* Point on "thinkx.net", so DNS issue with wrong IP?

We can see that "ada-lang.org" and "adalang.org" seem to be bought by the same person/entity, but these domains are not used, and not in sales. Looks like domain retention.

Does anyone know who is behind these domains?

From: Luke A. Guest <laguest@archeia.com>
Date: Thu, 24 Jun 2021 19:27:45 +0100

Steve is nerdboy on irc/github.

The Ada Lang one seems to be her name as there is a photo on the contact page.

There is learn/getadanow.com which is David Botton's.

Ada-related Tools

SweetAda 0.3 through 0.8

[Six consecutive announcements have been merged in a single thread. —arm]

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Subject: SweetAda 0.3 released
Date: Tue, 6 Apr 2021 09:09:52 -0700
Newsgroups: comp.lang.ada

I've just released SweetAda 0.3.

SweetAda is a lightweight development framework to create Ada systems on a wide range of machines. Please refer to https://www.sweetada.org.

First of all, please re-download toolchain packages; although timestamps do not change, they contain updated versions of GCC/GNAT wrappers which are essentials for a properly working build system which should have reached a stabilization point, and it seems rather efficient and free from major issues.

Release notes

- due to changes in RTSes, switch -gnatp (pragma Suppress (All_Checks)) is again commented out (Makefile.tc.in), to bring in exception processing; re-enable it if the final object is too big for your setup
- initial implementation of a secondary stack in the SFP RTS (not fully operational yet)
- strict compiler conformance, -gnatE and -gnato are defaults in Makefile.tc.in
- suppress No_Elaboration_Code in gnat.adc
- console.ali was not dragged in under GPRbuild mode and is missing in some configurations, which could lead to undefined references
- build.gpr/configure.gpr now correctly process implicit .ali units
- configure.ads (auto-generated from configuration.in) is pragma Pure
- a kernel link phase is performed if linker script changed
- Makefile now has two more targets: "session-start" and "session-end"; like "run" and "debug" targets, they are associated with shell commands that you can define in the platform configuration.in and can be useful for starting and ending JTAG servers, remote communication, and so on; these targets have nothing special, the names are only placeholders and their purposes are completely defined by the shell commands carried on; see an example in the new platform FRDM-KL46Z, where the commands respectively define an OpenOCD server startup and shutdown action
- Makefile.tc.in could specify -gsplit-dwarf; thus you can find *.dwo DWARF files in the object directory
- still more Makefiles tweaks: now there should be no problem building in GPRbuild mode; furthermore, "make createkernelcfg" forces a distclean
- menu-dialog.sh is standardized and behaves like menu.[bat|sh], so there are now separate items "createkernelcfg" and "configure" (previously there was a single "configure" item which executed them sequentially)
- elftool has a new command switch to find a symbol value: elftool -c findsymbol= which returns the symbol value; it is used, e.g., in the FRDM-KL46Z platform to automatically find the _start address in the executable image and instruct OpenOCD to run the target with a properly resume address; see an example as outlined in .../platforms/FRDM-KL46Z/runopenocd.tcl
- mbr can read other partitions beyond the first
- mbr partition read could cause a misaligned access with some CPUs, so an assignment is replaced with a memory copy
- Dreamcast code runs on a real device, not just in the GXemul emulator; this requires:
- a HIT-0400 "BroadBand Adapter" Ethernet module
- a CDI CD-ROM burned with "Dreamcast CDI Burner" https://alex-free.github.io/dcdib/
- a dc-tool-ip utility http://napalm-x.thegypsy.com/adk/dc/dcload-ip/index.html

  (the dc-tool-ip utility will be soon replaced by a Tcl script)
- MicroBlaze has now udivsi3 and umodsi3 LibGCC assembler routines
- MemecFX12 and Spartan3E platforms now have Tcl scripts to build, download and execute SweetAda kernel
- new target: FRDM-KL46Z Freescale/NXP ARM-CortexM0 board (a.k.a. "Freedom"), only able to blink a LED (needs OpenOCD to communicate with the target from inside SweetAda)
- FS-UAE platform renamed as Amiga-FS-UAE
- typos, cosmetics and minor adjustments

Quick notes

As usual, download the three packages core, RTS and LibGGC (since many changes are system-wide), and please save your work before overwriting the filesystem.

Subject: SweetAda 0.4 released
Date: Tue, 27 Apr 2021 03:53:57 -0700

I've just released SweetAda 0.4. [...]

Release notes

- SweetAda has a new toolchain, armeb-sweetada-eabi, to handle big-endian ARMs (previously this was not necessary since ZFP does not link against libraries); affected target is DigiConnectME — and eventually your own experimental target; ARM BE targets now not need to specify big-endian switches anymore, but they should explicitly specify armeb-sweetada-eabi as the toolchain name in the platform configuration.in, i.e.: TOOLCHAIN_NAME := $(TOOLCHAIN_NAME_ARMeb) whilst ARM LE takes the default toolchain
- the non-optimized versions of divsi3/modsi3 for MicroBlaze were not selected; this is now corrected
- the download script for Dreamcast — bba.tcl — is now written in Tcl (note: requires Tcl UDP extension) and thus does not require the dc-tool-ip utility, quick'n'dirty, no error checking yet; if it is difficult to install a Tcl extensions, then stuck yourself with dc-tool-ip by just uncommenting its exec line, and do an exit
- remove useless return statements in various Tcl scripts
- use Bits.BigEndian/LittleEndian booleans in llutils unit
- ATmega328P has more MCU definitions (not complete yet)
- ArduinoUno:
- XTAL clock frequency is specified in configure.ads
- ZFP profile is forced in configuration.in to avoid problems with a small foot-print memory, thus overriding the settings in the top-level configuration.in
- BSP does nothing; tests moved in application/test-ArduinoUno
- FRDM-KL46Z has more definitions; ZFP profile is forced in configuration.in to avoid problems with a small foot-print memory, thus overriding the settings in the top-level configuration.in
- bits.ads: some Bits_XX_Mask corrected; added Bits_XX_RMask
- now RTSes have, in every CPU target, two more files: 1) Makefile.srcs.in, the list of source files used (not of particular use so far, just a reference);

2) Makefile.rts.in, contains CPU-wide switches (i.e., not dependent on the multilib selected) used during the RTS build; those switches, which normally are empty, are automatically imported in the master Makefile and added to the target platform CPU, thus assuring that the compiler agrees on both RTS code and SweetAda/user code; as a consequence of this, there is no more need to specify, e,g, "-fno-leading-underscore" in SuperH/SH4 targets, and MIPS targets inherit automatically a -G0 switch (they are the only switches which are actually used in the RTS for those targets)

- QEMU-MIPS was based on "mips" machine; this machine does not exist anymore in current QEMU and so the platform is now based on "mipssim", what changes is just the UART16450 base I/O address in monitor.S
- Nios II Terasic DE10-Lite now has a Tcl front-end (programmer.tcl) which automatically downloads the SOF bitstream and executes the SweetAda code by means of Quartus command-line utilities and jtagd daemon
- Nios II Terasic DE10-Lite exposes a configuration setup that explains practically how to override core units, i.e., it invalidates last_chance_handler in the core directory and redefines the same subprogram package in its own directory, so to avoid dragging in the entire console package (which is used by the standard implementation of last_chance_handler)
- all Tcl scripts that handle the download of code on a physical target board are possibly renamed to a standardized "programmer.tcl"
- menu-dialog.sh now always shows warnings (previously it used to show warnings only if the build failed due to hard errors)
- typos, cosmetics and minor adjustments

Subject: SweetAda 0.5 released
Date: Tue, 4 May 2021 04:09:43 -0700

[...]

Release notes

- The SFP RTS now gets Ada.Tags installed, and so it should be possible to use Ada tagged types
- there are no more multiple Makefile.rts.in scattered in every multilib directory, only a single file is stored in the RTS root path of the toolchain target
- Master Makefile does not export FPU_MODEL, corrected
- new target: Synergy-S5D9 ARM-CortexM4 board, only able to blink a LED (needs OpenOCD to communicate with the target from inside SweetAda)
- LibGCC now has adddi3/subdi3/negdi2/mulsi3/muldi3 implemented in pure Ada (although a bit superfluous, since in most cases these subprograms will be overridden by CPU's own LibGCC assembly routines)
- The MVME162-510A platform has now a little Tcl script to download a SweetAda S-record image by means of 162-Bug on-board monitor communication; very simple script (and at 19200 also very slow for big images, but good enough for testing)
- the hard disk images for some platforms (Amiga-FS-UAE, Malta, PC-x86, etc) got accidentally deleted, they are now re-integrated for testing purposes
- removed superfluous conversion in Address_Displacement
- drivers/PC: PIC_Init has now Vector_Offset_Master/Slave input parameters and can be used also from non-x86 targets
- Malta MIPS: use PIC code from PC unit rather than an ad-hoc piece of code
- drivers/PC: PIT_Counter0_Init has an input Count parameter
- drivers/PC: unit does not depend on configure.ads anymore, and so the entire drivers branch should be CPU-independent
- typos, cosmetics and minor adjustments

Subject: ANN: SweetAda 0.6 released
Date: Wed, 19 May 2021 09:56:18 -0700

[...]

Release notes

- spurious entry core/last_chance_directory was not removed in the configuration.in for the core complex, and this causes a build failure in GPRbuild mode, corrected
- Makefile.tc.in: new ADAC_SWITCHES_WARNING switches: -gnatw.q - (Activate warnings on questionable layout of record types) -gnatw_r - (Activate warnings for out-of-order record representation clauses) (unused)
- Makefile.tc.in: added DISABLE_STACK_USAGE flag (some targets do not support stack usage computation, can be set from platform-level configuration.in)
- menu-dialog.sh remains in menu until you exit explicitly (e.g., by pressing double-ESC), so you can perform various actions sequentially; if instead you specify an action as an argument in the command line then the behaviour is unchanged, exiting at once after execution
- qemu-ifup.sh/qemu-ifdown.sh are now a single common copy in libutils directory; Dreamcast makeip.tcl/scramble.tcl are now merged in makecdrom.tcl; pc-x86-bootX.tcl moved as a single copy in share directory
- package Definitions is now placed in modules directory

- more error checking in various Tcl scripts
- initial cleanup of cpus branch file layout, removed duplicate files
- new target: SiFive HiFive1 Rev B, only able to blink the on-board RGB LEDs (needs OpenOCD to download the executable)
- Synergy-S5D9: bsp.ads got accidentally deleted, corrected
- Synergy-S5D9: added SCI definitions so that it can output something on SCI (UART mode, very primitive)
- partial rewriting of the NE2000 driver, more register definitions
- removed all ugly, unpleasant, ill-designed temporary code from exceptions.adb in PC-x86 interrupt handling (which now processes, e.g., raw TCP/IP traffic from applications.adb); the same in Amiga-FS-UAE
- some changes in Ethernet FIFO queue to make it more efficient
- ATmega328P (ArduinoUno): more register definitions, timers and general purpose registers; added some low-level templates; deleted unuseful subprogram in proprietary core unit and its dependency on console
- drivers/pc:
  - revised 8254 PIT; PIT_Counter0_Init now uses MODE 2 (rate generator) instead of MODE 3 (square wave generator) as a system timer
  - simple stub for RTC handling
  - IrqX renamed to PIC_IrqX
  - Irq0 aliased to PIT_Interrupt
  - Irq8 aliased to RTC_Interrupt
- added -mno-red-zone to GCC switches in x86-64
- use rounding instead of floor integer division when computing timing counts, where appropriate

There is also a new release of QEMU emulator — 20210517 — providing QEMU 6.0.0 for Linux and Windows platforms, and QEMU 5.2.0 for OS X. The OS X version should work on El Capitan (tested on a VM, someone reported problems on later versions).

Subject: ANN: SweetAda 0.7 released
Date: Tue, 1 Jun 2021 13:18:02 -0700

[...]

Release notes

- updated targets in master Makefile ("all" was tagged default instead of "help"); the targets "kernel_info" and "kernel_libinfo" are now exposed (kernel_libinfo produces listings of library objects even if the kernel build is not successful)
- added implicit dependencies for console unit
- elftool will emit spaces instead of TABs when performing an ELF section dump, this will be noted in the next toolchain release
- the linker script filename can now be declared in the platform configuration.in by specifying "LD_SCRIPT:= <linker_script_filename>", otherwise it takes a default "linker.lds"
- the C library now implements Ada stubs for malloc/free/calloc/realloc, so C code can call these Ada subprograms via stdlib wrappers; this has also the benefit of resolve references to malloc() when secondary stack tries to return heavy (i.e., unconstrained) objects, but be sure to add "USE_LIBGCC := Y" and "USE_CLIBRARY := Y" to the configuration.in file, either the generic one in the top-level directory, or the platform-dependent one
- SFP RTS: a-except: Raise_Exception calls Last_Chance_Handler
- SFP RTS: added Ada.Assertions (for pragma Assert you need to turn on -gnata in the "Ada Run-Time Checks switches" section of Makefile.tc.in)
- core/bits: added BITZERO/BITONE/BITL/BITH/BITOFF/BITON declarations
- core/console: Print (Boolean), emits "T" or "F"
- core/llutils: HexDigit_To_U8 uses a case instead of longer ifs
- modules/definitions: added a few definitions
- added various Volatile_Full_Access aspects here and there
- corrected some section wildcards in linker scripts for ARM platforms
- x86_64 lacks some low-level CPU subprograms (but they are empty anyway) and so the build could fail with unresolved objects, added
- new libutils/libopenocd.tcl file, useful for small OpenOCD function helpers
- Digi Connect ME (NET+ARM NS7520): some more register definitions; adopted a Tcl script as front-end to OpenOCD
- Synergy S5D9: OpenOCD cfg file renamed to standard "openocd.cfg"
- Synergy S5D9: more register definitions, SCI almost completely parameterized
- platform Spartan3E renamed as Spartan3E-SK
- new target: Avnet Xilinx Spartan-3A Evaluation Kit (Spartan3A-EK, MicroBlaze v7.00.b), only able to blink a LED; the programmer.tcl front-end will download the bitstream by directly interfacing with the on-board Cypress PSoC via USB protocol (no external tools needed in a Linux environment)
- targets involving OpenOCD (DigiConnectME, FRDM-KL46Z, HiFive1, MSP432P401R, STM32F769I, Synergy-S5D9) now should specify in the platform configuration.in the OpenOCD prefix (in Windows is the installation directory, i.e., that which is the parent of bin/, etc); the default is the *nix path "/usr/local"
- in the top-level directory there are the two files .cproject and .project for Eclipse CDT; no big deal since you have absolutely no Ada support, but if you import the project and configure the *.adb and *.ads files as textual source files, you could do a make build cycle, with error signalling (clicking on the error shown in console should redirect you to the offending source line)
- typos, cosmetics and minor adjustments

Subject: ANN: SweetAda 0.8 released
Date: Mon, 14 Jun 2021 07:46:15 -0700

[...]

Release notes @ https://www.sweetada.org/release_notes.html.

Downloads available @ https://sourceforge.net/projects/sweetada.

Release notes

- ADA_MODE defaults to ADA20 (-gnat2020)
- now the RTS is not a single common archive, instead every target CPU has its own — i.e. you have to download sweetada-rts-<target>-0.8.tar.gz; this way you avoid to waste bandwidth downloading a large RTS for, e.g., the AVR, when you don't need it
- adjusted Lock_Type definitions according to Ada 2020
- Tcl scripts: use "eq"/"ne" in place of "=="/"!="
- corrected a misinterpretation in the libopenocd.tcl proc version_numeric
- various programmer.tcl front-ends do not inherited OPENOCD_PREFIX, corrected
- change "adapter_khz" to "adapter speed" in OpenOCD configurations
- NiosII RTS has -mgpopt=none switch, so it is inherited in those kind of platforms
- NiosII lacks interrupt subprograms declaration (body still TBD), declared
- adjusted linker scripts in NiosII platforms (Altera10M50GHRD, DE10-Lite)
- DigiConnectME has ARM vectors template in llkernel.s
- adjusted some values in VGA low-level register programming; the package now

can setup graphic mode 12h (640x480x16)

- burned in an EPROM, SweetAda correctly startups in a DECstation 5000/133 and output messages on the SCC8530 serial port, blinking also the rear LEDs
- added/removed some Volatile_Full_Access aspects and cleaned up record layouts that don't need it (they are placed in the object definition instead)
- every CPUs has eventually an empty LibGCC package spec (which overrides thecore one); this enforces a catch of package (mis-)using, which cause the compiler to flag an error (should you try to use it when the CPU really does not need it)
- corrected a Makefile target dependency (output listings could be out-of-synch if "make postbuild" is not explicitly called)
- typos, cosmetics and minor adjustments

Quick notes

As usual, download the three packages core, RTS and LibGCC (since many changes are system-wide), and please save your work before overwriting the filesystem.

GNAT LLVM, ACATS

From: Simon Wright <simon@pushface.org>
Subject: GNAT LLVM, ACATS
Date: Wed, 07 Apr 2021 11:58:11 +0100
Newsgroups: comp.lang.ada

I recently had success building GNAT_LLVM on macOS: see notes here [1].

Running ACATS 4.1 U via the ACATS Grading tools as patched for llvm-gnat [2], I get impressively successful results: out of 4092 tests, GCC 11.0.1 of 2021-03-31 has

Result: Overall, B-Tests, C-Tests, L-Tests, Other Tests

[...]

Total Failed: 44, 30, 14, 0, 0

Total Not-Applicable: 35, 0, 35, 0, 0

Total Special: 182, 141, 21, 10, 10

Total Passed: 3831, 1272, 2420, 61, 78

(L-tests "check that all library unit dependencies within a program are satisfied before the program can be bound and executed, that circularity among units is detected, or that pragmas that apply to an entire partition are correctly processed". 'Special' means human inspection needed.)

whereas llvm-gnat, built from the same GCC sources, has

Result: Overall, B-Tests, C-Tests, L-Tests, Other Tests

[...]

Total Failed: 48, 31, 17, 0, 0

Total Not-Applicable: 35, 0, 35, 0, 0

Total Special: 184, 141, 23, 10, 10

Total Passed: 3825, 1271, 2415, 61, 78

[1] https://github.com/AdaCore/gnatllvm/issues/20#issuecomment-809400426

[2] https://github.com/simonjwright/ACATS-grading/tree/llvm

HAC V.0.095

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Subject: Ann: HAC v.0.095
Date: Wed, 7 Apr 2021 12:23:53 -0700
Newsgroups: comp.lang.ada

HAC (HAC Ada Compiler) is a small, quick, open-source Ada compiler, covering a subset of the Ada language. HAC is itself fully programmed in Ada.

Web site: http://hacadacompiler.sf.net/

Source repositories:

#1 svn: https://sf.net/p/hacadacompiler/code/HEAD/tree/trunk/

#2 git: https://github.com/zertovitch/hac

* Improvements since v.0.085:

Modularity: HAC recursively compiles all units needed to build a main program. Currently only procedures and functions bodies are supported - no packages yet, no separate specifications.

An example can be found in exm/unit_a.adb

Enjoy!

From: Stéphane Rivière <stef@genesix.fr>
Date: Thu, 8 Apr 2021 10:26:00 +0200

Well done Gautier!

In our company, we now use HAC here to replace many big Bash scripts in our servers cluster spread in 3 european DC. Big gain of productivity and maintainability.

Incidentally, HAC is up to 7 times faster than Bash for a much lower resource consumption. Surprising when you see the poverty of the syntax and semantics of Bash.

UXStrings 20210405

From: Blady <p.p11@orange.fr>
Subject: [ANN] UXStrings package available (UXS_20210405).
Date: Sun, 11 Apr 2021 10:45:53 +0200
Newsgroups: comp.lang.ada

A second POC implementation for UXStrings is provided. The source code files end with the number 2 as for instance "uxstrings2.ads".
https://github.com/Blady-Com/UXStrings/blob/master/src/uxstrings2.ads

A GNAT project file "uxstrings2.gpr" is provided with some naming conventions for both packages UXString and UXStrings.Text_IO.

Some API have been added to support ASCII 7 bits encoding for both version UXStrings 1 and 2. ASCII is a subset of UTF-8 thus no change with the internal UTF-8 representation.

However, in addition to UXStrings 1 implementation, the API is now aware if content is full ASCII. On one hand, this permits to access directly to the position of one character without iterating on UTF-8 characters. Thus this is a time improvement when content is full ASCII. On the other hand, when content is changing the API checks if the new content is full ASCII. Thus this is a time penalty when changes are not full ASCII.

English contents as programming text files are composed of lines in majority full ASCII but they may have some lines with characters out of the ASCII set. UXStrings is dealing with both.

Available on GitHub (https://github.com/Blady-Com/UXStrings) and also on Alire (https://alire.ada.dev/crates/uxstrings.html).

Feedback is welcome on the actual time improvement on your real use cases.

RAPID New Maintainer

From: Thomas <fantome.forums.tdecontes@free.fr.invalid>
Subject: RAPID
Date: Mon, 12 Apr 2021 18:56:52 +0200
Newsgroups: comp.lang.ada

Hi :-)

courtesy Oliver Kellogg, I'm officially the new RAPID maintainer :-)

I would like to know if there still exist some RAPID users :-)

I also would like to know if there are some users of other platforms than Unix or Windows who would like to use RAPID, even if it doesn't work until now.

I see that GtkAda supports at least Solaris/SPARC platform, in addition to Unix and Windows, but if no one is interested I won't waste time on a specific portability. Tell me :-)

RAPID maintainer

http://savannah.nongnu.org/projects/rapid/

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 12 Apr 2021 10:58:22 -0700

I'm on Windows and Solaris and Linux here, we might get Macintosh from long-term visitors.

From: Thomas <fantome.forums.tdecontes@free.fr.invalid>
Date: Mon, 12 Apr 2021 20:04:51 +0200

ok :-)

What's your relation with RAPID? (Are you a user? Are you interested? ...)

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 12 Apr 2021 13:01:00 -0700

Not a user, currently.

But interested, and having a nice cross-platform common-UI would make things a lot nicer for some prospective software-upgrades at work.

One such possible nicety would be a universal administration tool, another would be a data-management/-analysis tool for visiting scientists, another possibility would be decoupling several control-programs (codebases in everything from C to VB to C#) used to operate the instrumentation here from their host-systems and increase portability.

SparForte 2.4.1

From: Ken Burtch <koburtch@gmail.com>
Subject: ANN: SparForte 2.4.1 Released
Date: Sat, 1 May 2021 05:13:35 -0700
Newsgroups: comp.lang.ada

SparForte 2.4.1 has been released. This mainly contains fixes for the new tab completion system.

SparForte is my Ada-based shell, scripting language and web template engine.

SparForte can be downloaded at https://www.sparforte.com

The change log is located at https://www.sparforte.com/news/2021/news_may2021.html

SparForte is a hobby and relies on contributions from volunteers.

Simple Components v4.56

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components v4.56
Date: Sun, 2 May 2021 14:55:01 +0200
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), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, streams, multiple connections server/client designing tools and protocols implementations. The library is kept conform to the Ada 95, Ada 2005, Ada 2012 language standards.

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

Changes to the version 4.55:

- The packages Persistent.Streams and Persistent.Streams.Dump were added to implement streams backed by a fail-safe file;
- ELV/e-Q3 MAX! cube client was changed to work around cube firmware problems.

GCC 11.1.0 for MacOS

From: Simon Wright <simon@pushface.org>
Subject: ANN: GCC 11.1.0 for macOS
Date: Sun, 02 May 2021 17:28:09 +0100
Newsgroups: comp.lang.ada

GCC 11.1.0 x86_64-apple-darwin for macOS is available at: https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/11.1.0/native

The release no longer supports ASIS.

Libadalang and tools (gnatmetric, gnatpp, gnatstub, gnattest) are included.

Please note:

* This release is made as an installer package. Because I don't have a signing ID, you can't double-click on it; instead, right-button, Open, and ignore the warnings. Sorry.
* In the past, I've included modified versions of the compiler specs files. This time, such a modified specs file wouldn't run on El Capitan, so I've supplied the compiler as-built; if you're on Mojave or later, you need to set SDKROOT to pick up the place where Apple provides them. This means that the compiler won't automatically look in /usr/local/include (affects C, C++) or /usr/local/lib (affects all languages).

See the release notes at Sourceforge.

From: Simon Wright <simon@pushface.org>
Date: Thu, 10 Jun 2021 17:21:57 +0100

This release contains versions of gnatstub, gnattest, gnatpp and gnatmetric which fail to load:

$ /opt/gcc-11.1.0/bin/gnattest —help

dyld: Library not loaded: @rpath/libgnarl-11.dylib

Referenced from: /opt/gcc-11.1.0/bin/gnattest

Reason: image not found

Abort trap: 6

Workaround: export DYLD_FALLBACK_LIBRARY_PATH=/opt/gcc-11.1.0/lib/gcc/x86_64-apple-darwin15/11.1.0/adalib

You may not be aware that gprbuild now lets you specify building standalone static libraries "for Library_Interface use (list-of-units);"

- this doesn't work on macOS, and in fact cannot work, because it uses features of binutils object binaries that aren't available in Mach-O. The effect of this is that a static link against such a library will fail if the library involves any tasking. If you try to fix this by using the relocatable version, and then move the executable, it won't find the GNAT runtime dylibs.

I wonder why the GNAT runtime dylibs are all the way down there without a symlink in $prefix/lib?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 2 May 2021 18:54:25 +0200

Thanks Simon!

As a side note. The version 11 brings new incompatibilities breaking old code. In some cases X'Access is no longer accepted and needs to be replaced by X'Unchecked_Access.

I am too lazy to analyze whether that is a bug or feature, just be aware.

I dare say that every Ada style guideline should require 'Unchecked_Access everywhere. The issue became a permanent maintenance nightmare.

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 3 May 2021 10:29:38 +0200

> The release no longer supports ASIS.

That's unfortunate. Actually, GNAT FSF could be a good opportunity to continue support for ASIS.

> Libadalang and tools (gnatmetric, gnatpp, gnatstub, gnattest) are included.

But I guess not gnatcheck, since it needs ASIS.

From: Simon Wright <simon@pushface.org>
Date: Mon, 03 May 2021 12:14:11 +0100

> But I guess not gnatcheck, since it needs ASIS.

gnatcheck isn't in any of the recent releases of CE. [1], see the [Tools] section at the end, says it's not.

The [Tools] section also says ASIS is available as an add-on. But it says that about GNATtest.

[1] https://www.adacore.com/gnatpro/comparison

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 3 May 2021 11:46:51 +0100

> That's unfortunate. Actually, GNAT FSF could be a good opportunity to continue support for ASIS.

Why? ASIS is dead, even the WG don't bother anymore.

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 3 May 2021 13:50:38 +0200

> Why? ASIS is dead, even the WG don't bother anymore.

Many tools depend on ASIS, and there might well be an update of the standard. It is still supported by GnatPro (and PTC).

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Mon, 03 May 2021 16:16:50 +0100

> GCC 11.1.0 x86_64-apple-darwin for macOS is available at:

Thanks for that Simon.

My KDF9 emulator (~25KSLOC of Ada 2012) compiles and runs correctly, but the (stripped) object code is about 10% bigger and runs about 10% slower than a version compiled with GNAT CE 2020.

Is an Apple Silicon compiler a reasonable thing to hope for in the not too distant future? (Hint 8-)

From: Simon Wright <simon@pushface.org>
Date: Mon, 03 May 2021 16:44:36 +0100

Sorry to hear that.

> Is an Apple Silicon compiler a reasonable thing to hope for in the not too distant future? (Hint 8-)

Work is in progress, but not so far by me since I don't own any Apple Silicon :-(

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Tue, 04 May 2021 10:47:53 -0700

> I am too lazy to analyze whether that is a bug or feature, just be aware.

I had a similar issue upgrading from GNAT Community 2020 to GNAT Pro 21. The GNAT compiler has gotten smarter about enforcing accessibility rules.

Since those rules are there to prevent dangling references, they should be respected; I fixed my code to compile with 'Access.

It is a pain that GNAT didn't get this totally correct the first time around, but that's life.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 4 May 2021 22:12:42 +0200

> I had a similar issue upgrading from GNAT Community 2020 to GNAT Pro 21.

Alas.

> Since those rules are there to prevent dangling references, they should be respected; I fixed my code to compile with 'Access.

For example:

```
declare
   Location : Abstract_Layer'Class
renames
   Abstract_Layer'Class (Under.all);
begin
   ...
   if Location'Access =
      Location.Widget.Bottom then
```

This does not compile anymore. Clearly there cannot be any dangling references here.

Recent changes broke a lot of code involving comparisons of access types, especially if an anonymous access type is involved. Among them are cases when even 'Unchecked_Access does not help. So, one should resort to awful 'Address instead.

The situation is quite dire. I would even suggest introducing a built-in operation to compare an object with an access, e.g.

```
P'Refers (X) -- True if P points to X
```

since comparison of access types became too volatile.

> It is a pain that GNAT didn't get this totally correct the first time around, but that's life.

I am not a language lawyer to judge. My impression that in practice accessibility rules significantly reduce safety and code quality rather than add it.

AdaStudio-2021 Release 07/05/2021 Free Edition

From: Leonid Dulman <leonid.dulman@gmail.com>
Subject: Announce : AdaStudio-2021 release 07/05/2021 free edition
Date: Fri, 7 May 2021 00:05:38 -0700
Newsgroups: comp.lang.ada

I'm pleased to announce AdaStudio-2021 new release, based on Qt-6.1.0-everywhere extended with modules: qtconnectivity qtgraphicaleffect qtlocatio qtmultimedia qtsensors qtserialbus qtserialport qtwebchannel qtgamepad (qtscript and qtwebengine - work in progress).

Qt 6 is a new long-term project and I hope to solve these problems in next releases.

Qt6ada version 6.1.0 open source and qt6base.dll, qt6ext.dll (win64), libqt6base.so, libqt6txt.so(x86-64) built with Microsoft Visual Studio 2019 x64bin Windows, gcc x86-64 in Linux.

Package tested with GNAT gpl 2020 Ada compiler in Windows 64bit, Linux x86-64 Debian 10.4

Qt-6.1.0 everywhere opensource prebuilt binaries for win64 and x86-64 are included into AdaStudio-2021.

In new AdaStudio release was added new module qt6opencvada based on OpenCV 4.5.2 for camera capture, recording, transmit and receive.

qt6opencvada supports face detection and recognition. OpenCV-4.5.2 binaries prebuilt for win64 and x86-64 and included to AdaStudio.

AdaStudio-2021 includes the following modules: qt6ada,vtkada,qt6avada, qt6cvada and voice recognizer.

Qt6Ada is built under aGNU GPLv3alicense https://www.gnu.org/licenses/lgpl-3.0.html.

Qt6Ada modules for Windows, Linux (Unix) is available from Google drive https://drive.google.com/folderview?id=0B2QuZLoe-yiPbmNQRl83M1dTRVE&usp=sharing
(It can be mounted as virtual drive with ExpandDrive (https://www.expandrive.com/download-expandrive/)), go to Adastudio directory and load index.html to browser.

[Removed list of all file contents. —arm]

The full list of released classes is in "Qt6 classes to Qt6Ada packages relation table.docx"

If you have any problems or questions, let me know.

Leonid(leonid.dulman@gmail.com)

Gnoga 1.6a and 2.1-beta.

From: Blady <p.p11@orange.fr>
Subject: [ANN] Gnoga version 1.6a and 2.1-beta.
Date: Sat, 22 May 2021 10:07:46 +0200
Newsgroups: comp.lang.ada

Gnoga (https://sourceforge.net/projects/gnoga) version 1.6a has been released on SF GIT: https://sourceforge.net/p/gnoga/code/ci/dev_1.6/tree

Zipped source code is also available on: https://sourceforge.net/projects/gnoga/files

See HISTORY for details: https://sourceforge.net/p/gnoga/code/ci/dev_1.6/tree/HISTORY

Gnoga version V2.1-beta has been released on SF GIT branch dev_2.1: https://sourceforge.net/p/gnoga/code/ci/dev_2.1/tree

This version 2.1 is at the same functionality level as 1.6 with in addition the support of dynamic Unicode strings

via the UXStrings library (https://github.com/Blady-Com/UXStrings).

See HISTORY for details: https://sourceforge.net/p/gnoga/code/ci/dev_2.1/tree/HISTORY

V2.1 has been tested (demos, tests, tutorials) with GNAT Community 2020 on macOS 11.2 with Safari 14.

I propose that new features will be added only on this version.

Bug fixes will still be added on version 1.6.

Volunteers are welcome to test it further on their own configuration.

Some testing on Windows and Linux configuration will be appreciated.

Contributors are welcome.

Feel free to report detailed issues on Gnoga list or create tickets on SF:

https://sourceforge.net/p/gnoga/mailman/

https://sourceforge.net/p/gnoga/tickets/

Regards, Pascal.

https://blady.pagesperso-orange.fr

GNAT CE 2021

From: Adamagica <christ-usch.grein@t-online.de>
Subject: GNAT CE 2021 is out
Date: Thu, 27 May 2021 10:37:58 -0700
Newsgroups: comp.lang.ada

Just downloaded, no problems so far. Heureka.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Sat, 29 May 2021 10:17:12 -0700

There is a problem in the implementation of the new Reduce attribute. The Roman_Number example in RM 4.2.1(15/5ff) does not work. GNAT uses the wrong subtype for the Accum subtype. It's been reported.

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Wed, 2 Jun 2021 13:40:03 -0700

Regarding the Windows version: each time I call gprbuild on a project, everything is recompiled, in place of an incremental compilation (normally, only the Ada files that were modified since last build are recompiled). Did anyone else notice that?

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Tue, 08 Jun 2021 10:54:05 -0700

I see something similar but different (since 2019); no sources are recompiled, but everything is linked again, which in my builds is slow.

Assuming you are seeing the same thing, if you keep repeating the same build, it eventually finishes. If you look in *.bexch, after each build you will see one more gpr hash added; I gather they should all be added the first time. So the number of builds needed is the number of *.gpr you 'with', transitively. After that, each source code change only requires one build; changes to *.gpr (and some other things?) reset all the gpr hashes.

I have not reported this to AdaCore; I don't have access to a support contract for Windows. It would make sense to report it to the community channel.

From: Stephane Carrez <stephane.carrez@gmail.com>
Date: Fri, 11 Jun 2021 11:59:24 -0700

> Regarding the Windows version: each time I call gprbuild on a project, everything is recompiled [...]

I'm jumping at the end of your battle....

I've observed some small differences in the way compilation options are handled by gprbuild. In particular the Builder.Default_Switches and Compiler.Default_Switches were the source of a problem as far as I'm concerned. Not systematic but this resulted in your observed behavior.

From time to time I'm having the problem you mention and in most cases it was related to some incorrect definition in one of my GNAT projects. When this happens I use one of -vl or -vm options. And then... I lose a lot of time to spot the issue :-)

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Wed, 16 Jun 2021 09:24:37 -0700

> When this happens I use one of -vl or -vm options.

Actually my issue is very similar to one that appeared with GNAT GPL 2017 (see "GNAT GPL 2017 incremental compilation"). It is related to configuration pragma files. At the time the solution was to have in the .gpr project file compiler options expressed like

"-gnatec=" & project'Project_Dir & "debug.pra"

because gprbuild doesn't run in the same directory as GNAT since the GPL 2017 version. Now (four GPL/CE versions later) it's a bit trickier since gprbuild doesn't find the configuration pragma file at its correct place, even though GNAT does.

The verbosity switch (-vm) was helpful to confirm that (gprbuild shows reasons for recompilation). Thanks for the reminder!

PragmAda Reusable Components

From: Pragmada Software Engineering <pragmada@pragmada.x10hosting.com>
Subject: [Reminder] The PragmAda Reusable Components
Date: Tue, 1 Jun 2021 09:39:29 +0200
Newsgroups: comp.lang.ada

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

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

Archlinux 'gcc-ada-debug' AUR Package.

From: Rod Kay <rodakay5@gmail.com>
Subject: ANN: Archlinux 'gcc-ada-debug' AUR package.
Date: Fri, 11 Jun 2021 11:47:01 +1000
Newsgroups: comp.lang.ada

A gcc-ada-debug package has been added to the Archlinux AUR. It is identical to the official gcc-ada package except that debug symbols have not been stripped.

This allows for setting breakpoints on the standard exceptions of the Ada runtime library in GDB, as opposed to seeing a message such as 'Your Ada runtime appears to be missing debug information".

Ada Resource Embedder for C, Ada and Go

From: Stephane Carrez <stephane.carrez@gmail.com>
Subject: ANN: Ada Resource Embedder for C, Ada and Go
Date: Fri, 11 Jun 2021 06:30:42 -0700
Newsgroups: comp.lang.ada

I created a new tool to allow embedding any file in an Ada, C or Go binary. While embedding files, you can apply some transformations such as running a Javascript minifier (closure), compressing the file, encrypting it, ... The tool generates Ada, C or Go files that you compile with your program. In its simplest form, you can access the embedded content as a:

```
type Content_Access is access constant
Ada.Streams.Stream_Element_Array;
```

So the generated code only depends on Ada.Streams.

There are many modes that are explained in the documentation. For an overview, have a look at:

https://blog.vacs.fr/vacs/blogs/post.html?post=2021/06/11/Advanced-Resource-Embedder

And don't hesitate to fork, hack, and submit pull requests to:

https://github.com/stcarrez/resource-embedder

Well, for me it was a fun project :-)

Stephane

Ps: Go has a `go:embed` but It was fun to write the Go generator :-)

From: Stéphane Rivière <stef@genesix.org>
Date: Fri, 11 Jun 2021 15:51:52 +0200

Hi Stephane,

This is typically what I needed to improve my current AIDE v2 project (Ada Instant Development Environment - source GNAT CE 2019 2020 2021 - target Debian / Ubuntu with subtarget station (with GNATStudio, HAC, libs, debug aware RTS, and goodies) or server (bare minimal).

Many thanks!!!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 11 Jun 2021 18:19:15 +0200

I considered embedding into relocatable libraries similar to Windows' resource, e.g. for versioning plugins etc., but then decided to use an easier and more universal method.

I simply put an Ada constructing function into the library. The function is exported. The address returned by GetProcAddress or dlsym goes to Unchecked_Conversion to an access to subprogram, and here you are.

The obvious advantage of this method over embedding is that the object can be tagged of any derived type, which no embedding can do. And you can add whatever further initialization or checks you might need.

From: Stephane Carrez <stephane.carrez@gmail.com>
Date: Fri, 11 Jun 2021 10:32:56 -0700

Can you elaborate a little? I don't see what you put in your Ada constructing function. I do see how you use it but not how and where you put the original content or file.

Let's suppose you have some documentation file 'config/example.conf'. How would you integrate it in a binary with your solution?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 11 Jun 2021 20:25:55 +0200

> I don't see what you put in your Ada constructing function. I do see how you use it but not how and where you put the original content or file.

In my case I do not deal with files, it is always objects. In the simplest case it could be a record type like:

```
type Library_Data is record
   Do_This  : not null access procedure;
   Get_That : not null access function
              return That'Class;
end record;
```

That the library provides.

> Let's suppose you have some documentation file 'config/example.conf'.

The documentation would be an object ready for rendering. Say the documentation renderer is a GTK text view widget. Then that would require a text buffer:

```
type Library_Data is record
   Do_This : not null access procedure;
   Get_That : not null access function
              return That'Class;
   Documentation :  not null
                    Gtk_Text_Buffer;
end record;
```

The caller will drop the text buffer Documentation into a Gtk_Text_View widget to show the documentation.

I usually use programmatically generated content. E.g. HTML documentation would be a set of subprograms with parameters that put a portion of HTML into a stream/string. I then decorate their output as necessary, e.g. <tr>...</tr> if that must be a table cell etc.

The point is that documentation is almost never a static text, but has all sorts of parameters the provider does not know in advance.

From: Stephane Carrez <stephane.carrez@gmail.com>
Date: Fri, 11 Jun 2021 11:44:29 -0700

Thanks Dimitry for the clarification.

Different requirements lead to different solutions.

With ARE, I want to embed a Javascript file (such as jQuery), minify it with closure, compress it with gzip and make it available as raw content so that the web server can service it without reading any file. The content being accessible through either an Ada generated variable or through a function, it is mapped in memory (we avoid an open, read, close system call plus the gzip stuff) and the server only has to return the buffer content.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 11 Jun 2021 21:53:07 +0200

Same here. In the case of an integrated HTTP server I simply store the HTTP pages as a set of string constants and functions generating dynamic portions of. The HTTP server uses no external files. Most pages never exist in any moment of time, because the server generates portions of them and sends away chunks as soon as possible. For this reason I do not compress any pages. It would waste too much resources on a small embedded system and does not really matter for a large PC.

From: Stephane Carrez <stephane.carrez@gmail.com>
Date: Fri, 11 Jun 2021 23:15:30 -0700

> Same here. [...]

Not the same. ARE takes a static content and converts it in an Ada source that you compile.

Compression can help even on embedded systems because it reduces the size of data transfer. A jquery-3.4.1.js file is around 273K. Minified by closure it is reduced to 89K. If you also compress it, it reduces to 31K.

On slow networks these size reductions make a difference.

There is no waste of resources because the compression is made during the build process not at runtime.

I would say the opposite: what you get is smaller in size.

From: Stéphane Rivière <stef@genesix.org>
Date: Sat, 12 Jun 2021 14:03:41 +0200

> Your projects are very interesting, can I have the link for AIDE?

Github is coming - watch https://github.com/sowebio in july

Alpha is here https://stef.genesix.org/pub/ada/aide

Don't read v2.14 but v0.14 :) The v1 was a completely different beast. I just keep the numbering...

aide - binary

aide-2.14.zip - sources

v20-0.3.zip is the GP library used for AIDE

All this stuff is KISS[1] but very usable :)

sow - AIDE for Debian & Ubuntu Manual v34.pdf - read issues and to do list at the end. You should *wait for v2.15* release to test it, even the 'big view' already works. I have a few bugs to fix for a 'full flown experience' :)

sow - v20 Ada Library User Manual v28.pdf - API ref (a must read to

understand AIDE code) and more (methodology101 for kids and new coders). HAC Ada Compiler User Manual v82.pdf manual for HAC (HAC is included in AIDE. It's a very capable Ada subset interpreter, replacing Bash on all our servers cluster) HAC is a Gautier de Montmollin project (refs in doc). HAC is seven times faster than Bash, more productive and strongly typed!

[1] According to the "Keep It Simple Stupid" theory

GNAT CE 2021 for Intel MacOS

From: Simon Wright <simon@pushface.org>
Subject: ANN: GNAT CE 2021 for Intel macOS
Date: Thu, 17 Jun 2021 17:25:45 +0100
Newsgroups: comp.lang.ada

GNAT CE 2021, built for macOS El Capitan .. Big Sur, at Sourceforge: https://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2021-x86_64-darwin-bin/

Github (scroll down to the Assets section): https://github.com/simonjwright/distributing-gcc/releases/tag/gnat-ce-2021

Status of AdaControl (Cont.)

[See “Status of AdaControl” in AUJ 42-1, March 2021. —arm]

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Subject: Re: Status of AdaControl
Date: Sun, 9 May 2021 13:41:19 -0700
Newsgroups: comp.lang.ada

In the following blog post, you can find installation notes for using GNAT CE 2019 - just for the purpose of running AdaControl free edition!

https://gautiersblog.blogspot.com/2021/04/cleaning-up-hac-sources-with-adacontrol.html

A bit tedious, but doable. Hopefully the community will see again in the future, from time to time, updated snapshots with GNAT, ASIS and AdaControl "synchronized"...

The post also shows a demonstration of AdaControl "in action". Amazing tool!

Ada and Operating Systems

Ada Is Back on VMS

From: Vms Ada Alliance <contact@vmsadaall.org>
Subject: Ada is back on VMS
Date: Mon, 3 May 2021 17:42:26 +0200
Newsgroups: comp.lang.ada

Hello,

For whom the information is unknown, VMS itself is back since 2014. The company VSI (VMS Software Inc. Vmssoftware.com) negotiated with HP to give it support and future. Since this decision, VSI gives quality support to VMS on Alpha and Itanium hardware, and now the port to x86 and virtualization is about to be completed. Restricted Early Adopter Kits can be obtained, which run on Oracle Virtual Boxes or vmware. A full field test will begin in June, and the production release will be here at the end of this year.

For us, more important, the information is that Ada is back on VMS.

It had always been available on Alpha (and VAX) as DEC Ada, and is still there. It had been on Itanium until the end of 2014, as a GNAT Ada GCC implementation, supported by Adacore. AdaCore ended its support at the end of 2014, and we organized a rebuild from FSF sources, with the help of David Sauvage, AdaLabs. This build is presented here (https://github.com/AdaLabs/gnat-vms). On our side we can provide the binaries for users who can guarantee they have a professionnel or hobbyist licence for VMS (*) (the build uses VMS headers). (have a look here: http://www.vmsadaall.org/index.php/en/). The pia-sofer company (Play It Again SOFtwarE Renew, piasofer.fr) offers support for the package.

The novelty is that we can now think of a future of Ada on VMS, on x86. VSI organized its set of compilers using LLVM as the back end, version 10. And there is on github a prototype of a GNAT Ada front end for LLVM. So, we have just to test how to make them interact. It’s not a trivial project, but truly exciting.

We are just at the beginning of this work, and we’ll inform on this group how we are progressing.

For sure everyone is invited to participate. It’s a little bit difficult to get an Itanium, where you could test our build. Some itaniums are available via users club initiatives (not yet in France, but the users club vmsgenerations has a project on that). It’s easier to get a free Alpha emulator (for example here: http://www.migrationspecialties.com/FreeAXP.html). And you can get a hobbyist licence by VSI for both (https://vmssoftware.com/community/community-license). Important, we do think, to remember all the VMSisms before getting in the project. To test the GNAT Ada front end on Linux is straightforward, and it is also a preliminary work before porting on VMS (https://github.com/AdaCore/gnat-llvm). We don’t know when VSI will offer hobbyist licenses on VMS x86. Everyone who is thrilled to test some VMS x86 (and on Ada compiler) can contact us, and we’ll organize vpn accesses (contact@vmsadaall.org).

Our particular effort is about universality. We think it’s very important to address the cultural continuity of Ada on VMS from VAX to x86. There are a lot of good things written for Ada on VMS we have to reuse (for example for debug). And because the more modern solutions are around AdaCore solutions, it is important to understand similarities and differences between GNAT Ada on GCC and GNAT Ada on LLVM.

It’s a community effort. We hope VSI and/or AdaCore will get involved in the return of Ada on VMS. And we know there has always been a good collaboration between AdaCore and communities. But companies have their constraints, while individuals or Open Source structures can be immediately reactive. Perhaps also it is a sort of guarantee for users when a community is there.

Don’t hesitate to contact us: contact@vmsadaall.org

(*) It is a fullfledged 4.3.7 version, with all its GNAT tools. And we give in the same package the gcc C compiler and the gcc C++ compiler (of that version). We are working on the libstdc++-v3 (also here, every help welcomed, including just testing).

Ada and Other Languages

Pascal vs C Language Families

[Out of the blue a thread last seen in 2014 (!) was resurrected. It is an interesting yet mostly off-topic talk full of anecdotes about early compilers. —arm]

From: Paul Rubin <no.email@nospam.invalid>
Subject: Re: why the pascal family of languages (Pascal, Ada, Modula-2,2,Oberon, Delphi, Algol,...) failed compared to the C family?
Date: Thu, 27 May 2021 09:00:16 -0700
Newsgroups: comp.lang.ada

>>Algol 60 did not have a defined I/O.

> <?> Just curious -- do you mean the I/O was all by linked in

> function/subroutines rather than being keywords in the language?

Yeah, something like that. But there were successful Algol 60 implementations, including on Burroughs and Univac mainframes. C. A. R. Hoare supposedly called Algol 60 "a language so far ahead of its time, that it was not only an

improvement on its predecessors, but also on nearly all its successors.

>>I/O in Pascal was flawed.

> Well... It probably worked quite well in the original OS...

It wasn't just the I/O:

http://doc.cat-v.org/bell_labs/why_pascal/

Borland Turbo Pascal was very popular and apparently practical, though. I never used it but I have the impression that it (like most deployed Pascal implementations) somehow supplied workarounds to the limitations described in the paper above.

These were interesting:

* Things Turbo Pascal is Smaller Than:

https://prog21.dadgum.com/116.html

* Personal History of compilation speed part 2 (scroll down for the part about Turbo Pascal):

https://prog21.dadgum.com/47.html

The binary of Turbo Pascal was eventually released for no cost download, but apparently the source code was never released. That is disappointing based on how cool the above articles make it sound.

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Thu, 27 May 2021 13:53:29 -0400

> It wasn't just the I/O: http://doc.cat-v.org/bell_labs/why_pascal/

I believe the original goal for Pascal was to be a teaching language for algorithm development, and wasn't meant to be a real application programming language. Heck -- the earlier VMS Pascal required one to link into the FORTRAN libraries if one needed things like sin()/cos() functions.

> it (like most deployed Pascal implementations) somehow supplied workarounds to the limitations described in the paper above.

Which made it a "lock-in" language -- it wasn't Pascal as defined by Wirth and UCSD. One had to use compatible hardware (Windows, as I recall). Not much use when writing a satellite control (ground station) system on a VAX/Alpha machine (and yes, one such WAS written in VMS Pascal*... A few years later the new version was on HPUX [or whatever they called it] written in C -- they went from PDP-11 assembly to VAX Pascal to HP C)

>The binary of Turbo Pascal was eventually released for no cost download

I believe Turbo Pascal evolved into Borland's Delphi, which added OOP features. Now Embarcadero... And available in a "community edition" https://www.embarcadero.com/products/delphi/starter (interesting: they allow either Delphi OR C++Builder community editions, but not both on a computer [time for virtual machine images <G>]) Community license needs to be renewed annually (though is a free download as I read the site). OUCH -- Professional level is $1600 to start, and $400/year renewal (the initial $1600 is "perpetual", the $400/year is a subscription for updates/upgrades)

>disappointing based on how cool the above articles make it sound.

It feels bloated to me, but there is FreePascal with the Lazarus IDE.

My last Python exposure was on my TRS-80 model 4; Alcor Pascal (under a RatShack license name -- Model 3 was a pure Alcor release). It had the odd feature of allowing one to manually edit the "object" files. Once one learned the structure (they were ASCII) one could cut&paste functions/subroutines).

http://www.trs-80.org/alcor-pascal/

I seem to recall having Blaise II (editor) configured to work like VMS EDT (a bit of a trick, as the numeric pad only had 3 PF keys, not the 4 found on a VT100)

The realtime group did a survey of languages when they upgraded from the PDPs -- choices were VMS assembly, FORTRAN77, Pascal, and C. We had something like 30 people in the realtime group, and 70 people in the rest of the program skilled in F77. They tossed out C as error-prone, F77 as "old", assembly as "why change processor, then", and argued that many graduates at the time were learning Turbo Pascal in college.

When I saw the evaluation email, I sent back one that pointed out that Turbo Pascal had a lot of add-ons that would not be meaningful in a Wirth level Pascal (VMS did allow separate compilation, and linking to libraries written in other languages -- DEC had a common set of built-ins to define passing arguments as value/reference/descriptor so one could match the convention of the library language). I also pointed out that, having ignored the expertise of the 80+ F77 programmers, and gone the mile to choose Pascal, they might have fallen onto their faces and picked DEC VMS Ada -- a language designed for realtime processing...

A few years later, the department manager confessed that Pascal had been a mistake (I believe the lead realtime programmer had threatened to leave if Pascal was not picked -- manager caved in).

From: Wilson <winslowleon171@gmail.com>
Date: Thu, 27 May 2021 19:47:12 -0400

> Can we all blame this success of the C family of languages on Dennis Ritchie and Brian Kernighan brilliance and it being used for Unix? Another reason that C became so widespread was cost. In the 1970s many organizations (including universities) used the PDP 11 series of computers because they were cheap hardware.

Their OS for the PDP 11 on the other hand cost thousands of dollars per computer whereas C and UNIX were free. This was a bargain most owners found irresistible. Many schools made UNIX and C their standard educational language. This in turn generated graduates who only knew C putting pressure on employers to use C. Alas, it quickly became a self-feeding runaway success.

In short, a free lunch is hard to turn down no matter what comes with it.

From: John Perry <john.perry@usm.edu>
Date: Thu, 27 May 2021 17:34:07 -0700

> Borland Turbo Pascal was very popular and apparently practical, though.

I used Turbo Pascal in college 40 years ago, and yes! it did supply workarounds. Later I realized they looked a lot like features of Modula-2 (also by Wirth) and of Ada. Wikipedia tells me (And Therefore It Is True (TM) ;-)) that some of them come from UCSD Pascal.

This next paragraph is from memory, which may be corrupted, and I may have misunderstood it first, so don't take it too seriously, but: people who paid attention to what Wirth said and wrote about compiler design were able to produce small and fast compilers. Somewhere you can find a report written by one of Wirth's students about how they tried to modify one of their compilers to use a tree instead of a fixed-size array with linear search for the symbol table. Everyone except Wirth was sure that the tree would be both better and more useful, and everyone except Wirth turned out to be wrong. As I say, if it interests anyone I'm sure an online search will find it (but it might not be trivial, which is why I'm not doing it now myself).

> The binary of Turbo Pascal was eventually released for no cost download, but apparently the source code was never released. That is disappointing based on how cool the above articles make it sound.

FreePascal is an open-source reimplementation of Turbo Pascal. It boasts many of the speed advantages that Turbo Pascal has. I've never used it beyond occasionally downloading & playing with it, then forgetting about it.

From: Shark8 <onewingedshark@gmail.com>
Date: Fri, 28 May 2021 05:37:48 -0700

That story comes from the paper "Oberon: The Overlooked Jewel" — https://www.semanticscholar.org/paper/Oberon-The-Overlooked-Jewel-Franz/d48becdaf5c3d962e2778f804e8c64d292de408b—

> In order to find the optimal cost/benefit ratio, Wirth used a highly intuitive metric, the origin of which is unknown to me but that may very well be Wirth's own invention. He used the compiler's self-compilation speed as a measure of the compiler's quality. Considering that Wirth's compilers were written in the languages they compiled, and that compilers are substantial and non-trivial pieces of software in their own right, this introduced a highly practical benchmark that directly contested a compiler's complexity against its performance. Under the self-compilation speed benchmark, only those optimizations were allowed to be incorporated into a compiler that accelerated it by so much that the intrinsic cost of the new code addition was fully compensated.

> And true to his quest for simplicity, Wirth continuously kept improving his compilers according to this metric, even if this meant throwing away a perfectly workable, albeit more complex solution. I still vividly remember the day that Wirth decided to replace the elegant data structure used in the compiler's symbol table handler by a mundane linear list. In the original compiler, the objects in the symbol table had been sorted in a tree data structure (in identifier lexical order) for fast access, with a separate linear list representing their declaration order. One day Wirth decided that there really weren't enough objects in a typical scope to make the sorted tree cost-effective. All of us Ph.D. students were horrified: it had taken time to implement the sorted tree, the solution was elegant, and it worked well – so why would one want to throw it away and replace it by something simpler, and even worse, something as prosaic as a linear list? But of course, Wirth was right, and the simplified compiler was both smaller and faster than its predecessor.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 28 May 2021 15:28:48 +0200

I remember that Turbo Pascal had only one error message, something like "Syntax error in expression" with, God forbid, no column number.

Otherwise yes, it was pretty fast, much faster than MS-DOS C/C++ compilers of the time, Borland's own C++ including.

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Fri, 28 May 2021 07:49:44 -0700

> The binary of Turbo Pascal was eventually released for no cost download, but apparently the source code was never released.

Perhaps source code was not *officially* released but you find it easily on the Web (for TP 6.0) :-).

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Fri, 28 May 2021 08:01:10 -0700

Interestingly, the HAC Ada Compiler (https://hacadacompiler.sourceforge.io/, https://github.com/zertovitch/hac) is a distant descendent of Pascal-S by Wirth and identifiers are searched linearly via a linked list.

And yes, the compiler is very fast :-).

From: Paul Rubin <no.email@nospam.invalid>
Date: Fri, 28 May 2021 12:22:10 -0700

> Perhaps source code was not *officially* released but you find it easily on the Web (for TP 6.0) :-).

Very interesting I did find what I think you are referring to, though it is much larger than the old versions. It's weird that it looks like an official Borland release of some kind (complete with postal address for tech support). I didn't realize they had ever let that out. Anyway, thanks for the tip. The compiler proper seems to be about 25KLOC of almost entirely uncommented assembly code, plus Pascal headers. I will see if I can understand any of it. I wonder if it might be a disassembly, or otherwise obfuscated by Borland (by stripping comments) for the purpose of the release.

Sometime back I looked rather hard for the source code of TP (any version) and I found something claiming to be TP source code, but was actually something like a Windows wrapper. But this is different.

I think someone might have also published a disassembly of a released TP 2.0 binary, but I'm more interested in the original source as a historical artifact, rather than something to actually use and run in this day and age.

The comparable and amazing for the era BDS C was released as source code here: https://www.bdsoft.com/resources/bdsc.html

From: Luke A. Guest <laguest@archeia.com>
Date: Sun, 30 May 2021 17:12:50 +0100

> Otherwise yes, it was pretty fast, much faster than MS-DOS C/C++

Apparently, they were fast because the Turbo compilers didn't do optimisations due to the limitations of the machines.

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Sun, 30 May 2021 20:00:00 +0100

They were fast only by comparison with very slow compilers. I remember, around 1987, someone telling me in astonishment that Turbo ran at 2KSLOC/minute. I was unimpressed, as I had worked on a compiler that ran at 20KSLOC/min a decade earlier.

The 20KSLOC compiler ran on a 1.5MIPS machine.

From: Paul Rubin <no.email@nospam.invalid>
Date: Mon, 31 May 2021 20:34:18 -0700

Yes, but 1) 20KSLOC per what unit of time, 2) what language did it compile 1.5 mips is probably faster than a PC-XT (8088) but slower than a PC-AT (80286). I remember being impressed with the speed of Turbo C on the 386 that I used for a while. I never used Turbo Pascal. CP/M and the original PC were somewhat before my time.

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Tue, 01 Jun 2021 12:23:30 +0100

> Yes, but 1) 20KSLOC per what unit of time,

Per minute, as I said originally.

> 2) what language did it compile?

Pascal.

From: Paul Rubin <no.email@nospam.invalid>
Date: Tue, 01 Jun 2021 09:46:06 -0700

Ok, but that's maybe 5x slower than Turbo Pascal, which compiled 1000s of LOC per second on machines of that class.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 2 Jun 2021 16:38:59 +0200

> May I ask what is meant by "comparable machines"?

Around the end of 80's we used the dhrystone benchmark to compare machines.

> Here's why I ask: it can't have been a machine based on the Intel 8088, because that wasn't available until 1979. An elderly embedded engineer I know says that the CPU used in the PC series, at least the early PC's (8088 & 286) was a terrible CPU. He likes to joke how his <1MHz 6809-based "Trash 80 Color Computer" at $500 could run circles around the >4MHz 8088-based IBM at $1500.

That is my recollection too. I remember that an elderly 1MHz PDP-11 outperformed 12MHz 286. But the instruction set of PDP-11 was eons ahead of the 286's mess.

> So I'm curious if you know the basis of the claim that it was comparable hardware: clock speed, RAM, etc.It is simply that C compilers were garbage

> that time. C is a difficult language to compile compared to Turbo Pascal, especially using the methods that were popular then.

The only decent C compiler I remember from that time was DEC VAX C.

Even advanced machines like Motorola 68k, HP had horrific C compilers. (Sun's SPARC C was somewhat better.) The first thing to do was to bootstrap an early GCC on these machines, because the standard compilers were insufferable. I carried the sources with me on a tape (maybe even on a reel, I do not remember). That was fun.

From: Simon Wright <simon@pushface.org>
Date: Wed, 02 Jun 2021 18:26:10 +0100

> Even advanced machines like Motorola 68k, HP had horrific C compilers.

HP salesman: "Our compiler supports ANSI C (except for function prototypes)"

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Wed, 02 Jun 2021 19:13:06 +0100

> May I ask what is meant by "comparable machines"?

Machines that ran other programs at about the same speed, I specifically had in mind the Whetstone and the Ackermann function benchmarks.

> Here's why I ask: it can't have been a machine based on the Intel 8088, because that wasn't available until 1979.

The 20KSLOC/min compiler ran on an ICL 1906S, which had a Whetstone rating of ~800 KWIPS.

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Date: Wed, 2 Jun 2021 12:17:07 -0700

> But the Pascal family of languages (including Ada) have clearly failed to become popular, at least compared to the C-family (C, C++, C#, ...)

>

> The question is why did this happen?

By chance.

Most of the time a concept/idea is simply too ahead-of-time, or misunderstood.

In the 80s, there was P-code on the Apple II. It was beautiful and fast. But it was too early.

25 years later, Java bytecode came out, exactly the same thing, a cheeky clone.

It was a success (at least, commercially).

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 2 Jun 2021 18:11:10 -0500

P-code existed on a lot of machines; there was even a hardware CPU version of it. At least one of the early Ada compilers was built for as well.

> 25 years later, Java bytecode came out, exactly the same thing, a cheeky clone. It was a success (at least, commercially).

Given all of the above, p-code was a relative success as well. It just died out for whatever reason before Java came around doing approximately the same thing. (Quite possibly the possibility of practical just-in-time compilation made Java stick around longer than p-code.)

But the reality of it is that the hype machine got behind Java for whatever reason, but never really did behind Ada or p-code.

(Note that the intermediate code used by Janus/Ada was based on the ideas of p-code [with Ada-specific stuff like exception handling and tasking]; code at the level has a number of advantages. We never built an interpreter for it although it would be possible.)

From: Paul Rubin <no.email@nospam.invalid>
Date: Wed, 02 Jun 2021 16:40:52 -0700

In the 80s, there was P-code on the Apple II... 25 years later, Java bytecode came out, exactly the same thing, a cheeky clone.

The JVM isn't really comparable to P-code. It is a lot fancier, including features for multithreading and for garbage collection.

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Date: Wed, 2 Jun 2021 18:04:44 -0700

Well, obviously they put into the thing those mandatory features, plus 25 years of technological advantages (the P-code had to run on a 64kB, 1-MHz 6502), but the base concept is nearly the same, a stack-based VM.

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Date: Wed, 2 Jun 2021 18:29:09 -0700

> Given all of the above, p-code was a relative success as well.

I understand. No doubt that also the pcode was successful, but it is a pity that it couldn't make the point in the 90s, at least on microcomputer-class machines.

Still talking about the Apple II, I think one of the reasons is that it was way too complicated for the average user, with no less than 5 thick manuals and an underlying OS like the UCSD that was a radical departure in those times.

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Thu, 03 Jun 2021 12:51:19 -0400

> it is a pity that [p-code] couldn't make the point in the 90s, at least on microcomputer-class machines.

Faster processors, and the "standardization" on IBM's architecture, may have contributed... Since there was only "one" machine the portability of P-code (same object files, just provide a new interpreter layer for different architecture) wasn't a factor anymore.

> an underlying OS like the UCSD that was a radical departure in those times.

While the UCSD compiler was reasonable, the UCSD OS that went with it was a bit of a pain. All files had to be contiguous (no jumping to the next free sector), so one ended up periodically compressing the disk so files were at the front and all free-space consolidated. That also meant only one open output file per floppy drive, as output files opened in the largest contiguous free-space.

Ada Practice

Unreachable Branches in Case Statements

From: Reinert <reinkor@gmail.com>
Subject: Did I find a bug here?
Date: Thu, 1 Apr 2021 23:15:22 -0700
Newsgroups: comp.lang.ada

Assume this simple program:

```
procedure test0 is
  type A_Type is (A,B,C);
  subtype A_sub_Type is A_Type with
  Static_Predicate => A_sub_Type in A | B;
  X : A_type  := A;
  Y : A_sub_Type  := A;
begin
  case A_sub_Type(X) is
    when A => null;
    when B => null;
    when others => null;  -- ???? Should the
                          -- compiler complain here?
  end case;
end test0;
```

Should the compiler complain about "when others => null"? My compiler does not (running debian 10 updated, gnat-8).

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 2 Apr 2021 09:30:34 +0200

A case statement is allowed to have alternatives that cover no value. A friendly compiler can warn you that "this branch covers no value", but what you wrote is not illegal (and sometimes useful, if you have variants of your software that use slightly different definitions of the type).

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Fri, 2 Apr 2021 11:33:11 +0300

Recent discussion in ISO SC22 WG9, about the Ada part of the ISO

"programming language vulnerabilities" document, brought out that if the selecting expression (here AB_Type(X)) in a case statement or case expression has an invalid representation (for example, is an uninitialized variable with an out-of-range value), an Ada compiler is required to raise Constraint_Error if there is no "others" alternative, but if there is an "others" alternative the compiler can instead let execution proceed to that alternative without raising Constraint_Error.

In effect, "others" can cover all values, even those that are outside the nominal subtype of the selecting expression. See RM 5.4(12) and 5.4(13).

So if the programmer is worried about such cases (invalid representations from uninitialized variables or other causes such as Unchecked_Conversion), they can add an apparently unnecessary "others" alternative even if the other alternatives already cover all valid values. However, note that the compiler may choose to raise Constraint_Error even if there is an "others" alternative; RM 5.4 (10.d). To avoid that uncertainty, the program can perform an explicit 'Valid check before the case statement.

From: Reinert <reinkor@gmail.com>
Date: Fri, 2 Apr 2021 22:46:02 -0700

.....snip...

> values. However, note that the compiler may choose to raise

Could AB_Type(X) in "case AB_Type(X) is" function as such a valid check?

I try as much as possible to avoid "others" to make the compiler point out or to remind (in my large programs) where to add (or check for) possible alternatives in case I extend the value range of a variable. Then it may happen I need to put in for example something like "when A | B | C => null;" instead of "others".

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 3 Apr 2021 08:41:32 +0200

> Could AB_Type(X) in "case AB_Type(X) is" function as such a valid check?

Yes, but I recommend "case AB_Type'(X) is", i.e. a qualification rather than a conversion.

For the record:

A conversion carries the message: Take a value of type A and get the corresponding value from type B.

A qualification carries the message: I assume that the value belongs to (sub)type A.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 3 Apr 2021 11:18:47 +0300

> Le 03/04/2021 à 07:46, reinert a écrit :

>> Could AB_Type(X) in "case AB_Type(X) is" function as such a valid check?

> Yes,

I believe not. The use of X as an argument to a type conversion is an "evaluation" of X, by RM 4.6(28), which can be a bounded error by RM 13.9.1(9) if X'Valid is False.

That bounded error can lead to an exception or simply to continued execution with the invalid value.

> but I recommend "case AB_Type'(X) is", i.e. a qualification rather than a conversion.

That also requires an evaluation of X, by RM 4.7(4), and again can be a bounded error if X'Valid is False.

The use of X in X'Valid is explicitly defined to mean that X is "read" but is not "evaluated", RM 13.9.2(12)(13). So it seems to be the only safe way to check for validity.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 3 Apr 2021 14:37:08 +0200

> I believe not. The use of X as an argument to a type conversion is an "evaluation" of X [...]

Hmm, yes. I was thinking about eliminating "when others" because the intended range was covered. If you want to check for invalid values, 'Valid is the only way to go (that's why it was added to the language!)

Ada IRC Channel Migrates to Libera Chat

From: Rod Kay <rodakay5@gmail.com>
Subject: [ANN] Ada IRC channel on Freenode
Date: Wed, 7 Apr 2021 18:44:44 +1000
Newsgroups: comp.lang.ada

About twice a year we try to advertise the #ada channel on the Freenode IRC network. Celebrating its twentieth birthday this year, the channel continues to be active and friendly. These days it averages about 80 users at a time, large enough to support lively and informative discussions but small enough so it's not a madhouse.

Topics range all over the map, from building the latest GNAT to writing an OS in Ada to daily Ada programming issues to how to use PolyORB to use the Distributed Systems Annex. The stated topic is discussing Ada in the context of free and open-source software, but commercial users are equally welcome.

So fire up your favorite IRC client and come join us. The network is homed at irc.freenode.net, but has servers all over the world. Visit www.freenode.net on the web for details. Hope to see you soon!

From: John Mccabe <john@nospam.mccabe.org.uk>
Date: Wed, 7 Apr 2021 08:54:08 -0000

> About twice a year we try to advertise the #ada channel on the Freenode IRC network.

Wow! It's like a blast from the past; I've just started (re)using usenet and IRC's still around! Does anyone still use Gopher and Veronica? :-)

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Wed, 07 Apr 2021 11:39:01 -0400

> Does anyone still use Gopher and Veronica? :-)

I believe the Mother Gopher was killed off some years ago. However, there seem to be some 300 servers still out there based on links from Wikipedia.

From: Luke A. Guest <laguest@archeia.com>
Date: Wed, 7 Apr 2021 18:43:10 +0100

There are still gopher's out there, I think one might even be Ada related. Never heard of Veronica.

From: Jeffrey R. Carter
Date: Wed, 7 Apr 2021 20:24:11 +0200

Veronica was a successor to Jughead, which was a successor to Archie, a tool for indexing and searching FTP archives.

From: Shark8 <onewingedshark@gmail.com>
Date: Wed, 7 Apr 2021 12:01:16 -0700

> Does anyone still use Gopher and Veronica? :-)

I have a friend who does retro-computing and, IIRC, he's got a Gopher server running.

From: John Mccabe <john@nospam.mccabe.org.uk>
Date: Thu, 8 Apr 2021 14:28:34 -0000

> I have a friend who does retro-computing and, IIRC, he's got a Gopher server running.

Weirdo ;-)

From: Rod Kay <rodakay5@gmail.com>
Date: Thu, 27 May 2021 08:21:48 +1000

Due to a hostile, commercial takeover of Freenode, the #ada IRC channel has moved to the 'irc.libera.chat' server.

Different Public and Private Type Views

From: Drpi <314@drpi.fr>
Subject: GtkAda : Trying to derive a widget
Date: Thu, 8 Apr 2021 21:27:51 +0200
Newsgroups: comp.lang.ada

[The thread initially dealt with an ambiguity issue, but I have trimmed it

down to the part about type views which I find more relevant. —arm]

I'm trying to create a GtkAda widget derived from a standard widget.

```
-- debug_panel.ads
with Gtk.Scrolled_Window;
use Gtk.Scrolled_Window;
with Gtk.Text_View;
use Gtk.Text_View;
package Debug_Panel is
   type Debug_Panel_Record is new
   Gtk_Scrolled_Window_Record
   with private;
   -- [...]
private
   type Debug_Panel_Record is new
   Gtk_Scrolled_Window_Record
   with record
     Text : Gtk_Text_View;
   end record;
end Debug_Panel;
```

[...]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 9 Apr 2021 00:27:26 +0200

[...]

P.S. When you create new widget it is better to use a more general ancestor hiding insufficient details, e.g.

```
   type Debug_Panel_Record is
     new Gtk.Widget.Gtk_Widget_Record
     with private;
   ...
private
   type Debug_Panel_Record is
     new Gtk.Scrolled_Window.
     Gtk_Scrolled_Window_Record with
   record
     ...
   end record;
```

This makes the code less fragile if you later decide to choose another container widget for the base.

From: Drpi <314@drpi.fr>
Date: Fri, 9 Apr 2021 07:28:46 +0200

> P.S. When you create new widget it is better to use a more general ancestor hiding insufficient details

I'm surprised it is possible to write such a thing in Ada. What does the compiler do with this?

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 9 Apr 2021 08:12:19 +0200

> I'm surprised it is possible to write such a thing in Ada.

There is no problem, since Gtk.Scrolled_Window. Gtk_Scrolled_Window_Record is a descendant of Gtk.Widget.Gtk_Widget_Record. There is no lie: a Debug_Panel_Record IS A Gtk_Widget_Record. The private view has more information: it actually IS A Gtk_Scrolled_Window_Record, but the extra properties are not accessible outside from the package body.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 9 Apr 2021 08:18:28 +0200

> I'm surprised it is possible to write such a thing in Ada.

The public view is Gtk_Widget_Record [with no record members], the full view is Gtk_Scrolled_Window_Record [with record members you specified].

From: Drpi <314@drpi.fr>
Date: Fri, 9 Apr 2021 13:32:03 +0200

>There is no lie: a Debug_Panel_Record IS A Gtk_Widget_Record. The private view has more information: it actually IS A Gtk_Scrolled_Window_Record, but the extra properties are not accessible outside from the package body.

Ok. That's interesting. One more thing learned today :)

Importing Types at Run Time

From: Daniel Norte Moraes <danielcheagle@gmail.com>
Subject:How get/use a 'type' from a Ada shared library loaded at run time (plugin)?
Date: Thu, 15 Apr 2021 01:48:17 -0700
Newsgroups: comp.lang.ada

Hi All!

How to use Ada 'type(s)' from a library loaded at run time (dlopen &Cia)? Are they possible?

My main need is to declare variables from these types or and make dispatching calls based on these types.

Any help is welcome.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 15 Apr 2021 11:28:24 +0200

> How to use Ada 'type(s)' from a library loaded at run time (dlopen &Cia)? Are they possible?

Like any other type.

You should not forget to initialize the library though. The library project should normally have

```
for Library_Auto_Init use "false";
```

because otherwise it would likely deadlock under Windows [if you wish to make your project portable]. Under Linux I am not sure if it deadlocks, never tested for that.

> My main need is to declare variables from these types or and make dispatching calls based on these types.

You cannot declare variables of types declared in a dynamically loaded library for the obvious reason that you cannot dynamically refer to a package from the library.

But you can dispatch on a class-wide object which specific type is declared in a dynamically loaded library.

For example you can call a dynamically loaded constructing function that returns T'Class where T is declared outside the library and then dispatch to an overridden primitive operation of S derived from T inside the library.

To my understanding library initialization expands dispatching tables with all tagged types declared in the library.

P.S. What happens on finalization is an intriguing question. I would rather never attempt to unload an Ada library...

Official Ada Logo/Icon

From: Heziode <heziode@protonmail.com>
Subject: Does Ada have an official logo/icon?
Date: Thu, 15 Apr 2021 14:43:20 +0200
Newsgroups: comp.lang.ada

By reading an article, I visited the official website of JWT (JavaScript Web Token): https://jwt.io

On this website, they list implementations in different languages, particularly Ada. However, I have never seen this logo for Ada language.

So, the question is: Does Ada have an official logo/icon?

After some research, there does not seem to be a "really official" logo, but just recommendation (see http://getadanow.com/#mascot)

Unchecked_Deallocation Usefulness

[Although the original post addressed another issue, I have trimmed the thread to a particular side topic about the usefulness of Unchecked_Deallocation. —arm]

From: Drpi <314@drpi.fr>
Subject: Unchecked_Deallocation with tagged types
Date: Sat, 17 Apr 2021 23:45:27 +0200
Newsgroups: comp.lang.ada

I have the following types :

```
type t_Element_Record is tagged null
record;
type t_Str_Record (Str_Length : Natural)
is new t_Element_Record with private;
```

Do I have to create a Unchecked_Deallocation procedure for each tagged type or only one for the root tagged type (and the compiler manages the effective tagged type)?

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Sun, 18 Apr 2021 01:46:07 -0700

Side note: did anyone already suggest a new keyword: unchecked_free and a special statement:

```
unchecked_free Some_Pointer;
```

?

From: Jeffrey R. Carter
Date: Sun, 18 Apr 2021 11:09:59 +0200

> unchecked_free Some_Pointer;

For every access variable P, there could exist the attribute procedure

```
P'Free;
```

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 18 Apr 2021 12:13:25 +0200

> For every access variable P, there could exist the attribute procedure
>
> P'Free;

I like the idea of attaching it to a variable rather than to type.

I remember the claim that originally making it a generic procedure with an indigestible name was meant as a barrier for lazy programmers. Plus some considerations regarding garbage collection lurked in the subconscious.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 18 Apr 2021 12:20:56 +0200

> P'Free;

Which would defeat the goal of Unchecked_Deallocation.

Ada (or more precisely Ichbiah) chose to deallocate through an instantiation of a generic for good reason. Deallocation is a place where many problems can come from, it is important to be able to trace where they are. The current solution forces you to put "with Unchecked_Deallocation" on top of every module that deallocates, therefore telling the reader that there is some danger in it, and making it easier to find where all the deallocations happen.

Now, I know that in those days, ease of writing is considered more important than ease of reading and long term maintenance...

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 18 Apr 2021 12:34:39 +0200

> [...]The current solution forces you to put "with Unchecked_Deallocation" on top of every module that deallocates, [...]

No, that does not work. If we are supposed to search for all calls to deallocate, then attribute 'Free is much easier to look after than first looking for "with Unchecked_Deallocation", then for an instantiation of it with the types in question and then for the name of the instance.

> Now, I know that in those days, ease of writing is considered more important than ease of reading and long term maintenance...

Yes, but dangling pointers is a more complex problem, that effortlessly passes brief code inspections. I agree with you in general, but disagree in this case. *IF* pointers are used *THEN* Unchecked_Deallocation only obfuscates things rather than helps.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 18 Apr 2021 17:14:12 +0200

> [...] much easier to look after than first looking for "with Unchecked_Deallocation"

meant it the other way round: if the module has no "with unchecked_deallocation", you know it does not deallocate anything, without looking at the entire body.

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Sun, 18 Apr 2021 08:23:40 -0700

Not at all: the instantiation can be defined in another package - and it is often the case - with any name (Free, Dispose, ...). So actually with the present way it is difficult to track where unchecked deallocation is used, plus it is tedious for the programmers. The P'Free attribute or the "unchecked_free P;" statement would be straightforward to track.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 18 Apr 2021 17:53:11 +0200

Well, P'Free can also be in another package... Of course, we are talking here only about the direct, actual deallocation.

If you want to precisely know where deallocation is used, use AdaControl (for any solution). If you want to be confident that there is no direct deallocation in a module, the generic wins.

And after all, the attribute only saves you one line of code... (OK, two if you count the "with" ;-))

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 20 Apr 2021 21:35:59 +0200

> OTOH, an Ada follow-on would most likely have access types with automatic deallocation as proposed by Tucker in one of the many AIs on ownership. So using any form of explicit deallocation would be discouraged (as would the use of raw pointer types).

I do not understand how that could work, it sounds like a halting problem to me, but anyway, where is a problem? Add a whole new hierarchy of access types independent of the existing one.

From: Jeffrey R. Carter
Date: Tue, 20 Apr 2021 22:32:16 +0200

> 'Free makes more sense in a new language (an Ada follow-on).

Right. I don't think it would be a good idea to add it to Ada.

But I think a new language should not have pointers at all.

No more radical than not having arrays.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Wed, 21 Apr 2021 00:10:40 +0300

> But I think a new language should not have pointers at all. No more radical than not having arrays.

It seems to me that a language without arrays and pointers would be very difficult to use in an embedded, real-time, close-to-HW context. So we would lose the nice wide-spectrum nature of Ada.

From: Jeffrey R. Carter
Date: Wed, 21 Apr 2021 10:35:45 +0200

I don't see that pointers are needed for such S/W.

Brukardt has recently been discussing the idea that a high-level language such as Ada should not have arrays, which is why I referenced it. Such a language might not be convenient for such systems.

But the idea is that arrays are a low-level implementation feature that are usually used to implement higher-level abstractions, such as sequences and maps. A language without arrays would have direct support for such abstractions. My experience is that most uses of arrays in embedded, real-time S/W are also for such abstractions, so it would probably not be too great a problem.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 21 Apr 2021 12:11:07 +0200

> I don't see that pointers are needed for such S/W.

Try to load and bind a relocatable library without pointers.

> A language without arrays would have direct support for such abstractions.

That is not enough, even if providing such abstractions were viable. Which is not, because they would be far more complex than array abstraction and resolve none of the problems array abstraction has. E.g. container subtypes constrained to subtypes of elements and/or subtypes of keys.

Array is a simplest case of container. If you cannot handle arrays, how do you hope to handle maps?

Then see above, and explain how an opaque map will deal with a shared memory mapped into the process address

space? Or what would be the primitive operation Write of Root_Stream_Type?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 23 Apr 2021 19:49:24 -0500

> It seems to me that a language without arrays and pointers would be very difficult to use in an embedded, real-time, close-to-HW context.

It's important that a new language have a way to interface to existing hardware and software. So there has to be something that maps to C arrays and pointers (and the equivalent for hardware). But that doesn't necessarily have to be something that is used outside of interfacing. An Ada example is Unchecked_Unions -- they exist for interfacing but shouldn't be used otherwise. A fixed vector type and a raw general access type would do the trick, but those could be something that are almost never used outside of interfacing packages.

Ada and Unicode

From: Drpi <314@drpi.fr>
Subject: Ada and Unicode
Date: Sun, 18 Apr 2021 00:03:11 +0200
Newsgroups: comp.lang.ada

Hi,

I have a good knowledge of Unicode: code points, encoding... What I don't understand is how to manage Unicode strings with Ada. I've read part of ARM and did some tests without success.

I managed to be partly successful with source code encoded in Latin-1. Any other encoding failed. Any way to use source code encoded in UTF-8? In some languages, it is possible to set a tag at the beginning of the source file to direct the compiler which encoding to use. I wasn't successful using -gnatW8 switch. But maybe I made too many tests and my brain was scrambled.

Even with source code encoded in Latin-1, I've not been able to manage Unicode strings correctly.

What's the way to manage Unicode correctly?

From: Luke A. Guest <laguest@archeia.com>
Date: Sun, 18 Apr 2021 01:02:06 +0100

It's a mess imo. I've complained about it before. The official stance is that the standard defines that a compiler should accept the ISO equivalent of Unicode and that a compiler should implement a flawed system, especially UTF-8 types, http://www.ada-auth.org/standards/rm12_w_tc1/html/RM-A-4-11.html

Unicode is a bit painful, I've messed about with it to some degree here: https://github.com/Lucretia/uca.

There are other attempts:

1. http://www.dmitry-kazakov.de/ada/strings_edit.htm
2. https://github.com/reznikmm/matreshka (very heavy, many layers)
3. https://github.com/Blady-Com/UXStrings

I remember getting an exception converting from my unicode_string to a wide_wide string for some reason ages ago.

From: Maxim Reznik <reznikmm@gmail.com>
Date: Mon, 19 Apr 2021 01:29:35 -0700

> Any way to use source code encoded in UTF-8?

Yes, with GNAT just use "-gnatW8" for compiler flag (in command line or your project file):

-- *main.adb:*

```
with Ada.Wide_Wide_Text_IO;
procedure Main is
   Привет : constant Wide_Wide_String :=
              "Привет";
begin
   Ada.Wide_Wide_Text_IO.Put_Line
   (Привет);
end Main;
```

```
$ gprbuild -gnatW8 main.adb
$ ./main
Привет
```

> In some languages, it is possible to set a tag at the beginning of the source file to direct the compiler which encoding to use.

You can do this by putting the Wide_Character_Encoding pragma (This is a GNAT specific pragma) at the top of the file. Take a look:

-- *main.adb:*

```
pragma Wide_Character_Encoding (UTF8);
with Ada.Wide_Wide_Text_IO;
procedure Main is
   Привет : constant Wide_Wide_String :=
              "Привет";
begin
   Ada.Wide_Wide_Text_IO.Put_Line
   (Привет);
end Main;
```

```
$ gprbuild main.adb
$ ./main
Привет
```

> What's the way to manage Unicode correctly?

You can use Wide_Wide_String and Unbounded_Wide_Wide_String type to process Unicode strings. But this is not very handy. I use the Matreshka library for Unicode strings. It has a lot of features (regexp, string vectors, XML, JSON, databases, Web Servlets, template engine, etc.).

URL: https://forge.ada-ru.org/matreshka

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Mon, 19 Apr 2021 02:08:35 -0700

> Any way to use source code encoded in UTF-8 ?

```
for Switches ("non_ascii.ads")
use ("-gnatiw", "-gnatW8");
```

from the GNAT user guide, 4.3.1 Alphabetical List of All Switches:

`-gnati`c"

Identifier character set (`c' = 1/2/3/4/8/9/p/f/n/w). For details of the possible selections for `c', see *note Character Set Control: 4e.

This applies to identifiers in the source code

`-gnatW`e"

Wide character encoding method (`e'=n/h/u/s/e/8).

This applies to string and character literals.

> What's the way to manage Unicode correctly?

There are two issues: Unicode in source code, that the compiler must understand, and Unicode in strings, that your program must understand.

(I've never written a program that dealt with utf strings other than file names).

-gnati8 tells the compiler that the source code uses utf-8 encoding.

-gnatW8 tells the compiler that string literals use utf-8 encoding.

package Ada.Strings.UTF_Encoding provides some facilities for dealing with utf. It does _not_ provide walking a string by code point, which would seem necessary.

We could be more helpful if you show what you are trying to do, you've tried, and what errors you got.

From: Drpi <314@drpi.fr>
Date: Mon, 19 Apr 2021 11:28:34 +0200

> pragma Wide_Character_Encoding (UTF8);

Wide and Wide_Wide characters and UTF-8 are two distinct things. Wide and Wide_Wide characters are supposed to contain Unicode code points (Unicode characters). UTF-8 is a stream of bytes, the encoding of Wide or Wide_Wide characters. What's the purpose of "pragma Wide_Character_Encoding (UTF8);"?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 19 Apr 2021 11:34:26 +0200

> -gnati8 tells the compiler that the source code uses utf-8 encoding.

>

> -gnatW8 tells the compiler that string literals use utf-8 encoding.

Both are recipes for disaster, especially the second. IMO the source must be strictly ASCII 7-bit. It is less dangerous to have UTF-8 or Latin-1 identifiers, they could be at least checked, except when used for external names. But string literals would be a ticking bomb.

If you need a wider set than ASCII, use named constants and integer literals. E.g.

```
Celsius : constant String := Character'Val (16#C2#) & Character'Val (16#B0#) & 'C';
```

From: Simon Wright <simon@pushface.org>
Date: Mon, 19 Apr 2021 12:15:29 +0100

> воскресенье, 18 апреля 2021 г. в 01:03:14 UTC+3, DrPi:

>> Any way to use source code encoded in UTF-8?

> Yes, with GNAT just use "-gnatW8" for compiler flag

But don't use unit names containing international characters, at any rate if you're (interested in compiling on) Windows or macOS:

https://gcc.gnu.org/bugzilla/show_bug.cgi?id=81114

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 19 Apr 2021 12:50:40 +0100

> But don't use unit names containing international characters,

There's no such thing as "character" any more and we need to move away from that. Unicode has the concept of a code point which is 32 bit and any "character" as we know it, or glyph, can consist of multiple code points.

In my lib, nowhere near ready (whether it will be I don't know), I define octets, Unicode_String (utf-8 string) which is an array of octets and Code_Points which an iterator produces as it iterates over those strings. I was intending to have an iterator for grapheme clusters and other units.

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 19 Apr 2021 12:56:34 +0100

> There are two issues: Unicode in source code, that the compiler must understand, and Unicode in strings, that your program must understand.

And this is where the Ada standard gets it wrong, in the encodings package re utf-8.

Unicode is a superset of 7-bit ASCII not Latin 1. The high bit in the leading octet indicates whether there are trailing octets. See https://github.com/Lucretia/uca/blob/master/src/uca.ads#L70 for the data layout. The first 128 "characters" in Unicode match that of 7-bit ASCII, not 8-bit ASCII, and certainly not Latin 1. Therefore this:

```
package Ada.Strings.UTF_Encoding
   ...
   subtype UTF_8_String is String;
   ...
end Ada.Strings.UTF_Encoding;
```

Was absolutely and totally wrong.

...and, before someone comes back with "but all the upper half of latin 1" are represented and have the same values." Yes, they do, in Code points which is a 32 bit number. In UTF-8 they are encoded as 2 octets!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 19 Apr 2021 14:52:43 +0200

> subtype UTF_8_String is String;

> Was absolutely and totally wrong.

It is a practical solution. Ada type system cannot express differently represented/constrained string/array/vector subtypes. Ignoring Latin-1 and using String as if it were an array of octets is the best available solution.

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 19 Apr 2021 14:00:39 +0100

They're different types and should be incompatible, because, well, they are. What does Ada have that allows for this that other languages don't? Oh yeah! Types!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 19 Apr 2021 15:10:49 +0200

They are subtypes, differently constrained, like Positive and Integer. Operations are the same, values are differently constrained. It does not make sense to consider ASCII 'a', Latin-1 'a', UTF-8 'a' different. It is the same glyph differently encoded. Encoding is a representation aspect, ergo out of the interface!

BTW, subtype is a type.

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 19 Apr 2021 14:15:24 +0100

> They are subtypes, differently constrained, like Positive and Integer.

No they're not. They're subtypes only and therefore compatible. The UTF string isn't constrained in any other ways.

> It is the same glyph differently encoded.

As I already said in Unicode the glyph is not part of Unicode. The single code point character concept doesn't exist anymore.

> BTW, subtype is a type.

subtype is a compatible type.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 19 Apr 2021 15:31:28 +0200

> [...]

> subtype is a compatible type.

Ada subtype is both a sub- and supertype, i.e. substitutable [or so the compiler thinks] in both directions. A derived tagged type is substitutable in only one direction.

Neither is fully "compatible", because otherwise there would be no reason to have an exactly same thing.

From: Vadim Godunko <vgodunko@gmail.com>
Date: Mon, 19 Apr 2021 06:18:22 -0700

> What's the way to manage Unicode correctly?

Ada doesn't have good Unicode support. :(So, you need to find a suitable set of "workarounds".

There are few different aspects of Unicode support need to be considered:

1. Representation of string literals. If you want to use non-ASCII characters in source code, you need to use -gnatW8 switch and it will require use of Wide_Wide_String everywhere.
2. Internal representation during application execution. You are forced to use Wide_Wide_String at the previous step, so it will be UCS4/UTF32.
3. Text encoding/decoding on input/output operations. GNAT allows to use UTF-8 by providing some magic string for Form parameter of Text_IO.

It is hard to say that it is a reasonable set of features for the modern world. To fix some of the drawbacks of the current situation we are developing a new text processing library, known as VSS.

https://github.com/AdaCore/VSS

At the current stage it provides encoding independent API for text manipulation, encoders and decoders API for I/O, and JSON reader/writer; regexp support should come soon.

Encoding independent API means that applications always use Unicode characters to process text, independently from the real encoding used to store information in memory (UTF-8 is used for now, UTF-16 will be added later for interoperability with Windows API and WASM). Coders and encoders allow translation from/to different encodings when applications exchange information with the world.

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 19 Apr 2021 15:24:36 +0200

> They're different types and should be incompatible, because, well, they are.

They are not so different. For example, you may read the first line of a file in a string, then discover that it starts with a BOM, and thus decide it is UTF-8.

BTW, the very first version of this AI had different types, but the ARG felt that it would just complicate the interface for the sake of abusive "purity".

From: Maxim Reznik <reznikmm@gmail.com>
Date: Mon, 19 Apr 2021 06:50:42 -0700

What's the purpose of "pragma Wide_Character_Encoding (UTF8);"?

This pragma specifies the character encoding to be used in program source text...

https://docs.adacore.com/gnat_rm-docs/html/gnat_rm/gnat_rm/implementation_defined_pragmas.html#pragma-wide-character-encoding

I would suggest also this article to read:

https://two-wrongs.com/unicode-strings-in-ada-2012

From: Drpi <314@drpi.fr>
Date: Mon, 19 Apr 2021 17:48:18 +0200

> Code points which is a 32 bit number. In UTF-8 they are encoded as 2 octets!

A code point has no size. Like universal integers in Ada.

From: Drpi <314@drpi.fr>
Date: Mon, 19 Apr 2021 18:07:32 +0200

> They're different types and should be incompatible, because, well, they are

I agree.

In Python2, encoded and "decoded" strings are of same type "str". Bad design.

In Python3, "decoded" strings are of type "str" and encoded strings are of type "bytes" (byte array). Both are different things and can't be assigned one to the other. Much more clear for the programmer. It should be the same in Ada. Different types.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 20 Apr 2021 14:06:26 -0500

> They're different types and should be incompatible

If they're incompatible, you need an automatic way to convert between representations, since these are all views of the same thing (an abstract string type). You really don't want 35 versions of Open each taking a different string type.

It's the fact that Ada can't do this that makes Unbounded_Strings unusable (well, barely usable). Ada 202x fixes the literal problem at least, but we'd have to completely abandon Unbounded_Strings and use a different library design in order for it to allow literals. And if you're going to do that, you might as well do something about UTF-8 as well -- but now you're going to need even more conversions. Yuck.

I think the only true solution here would be based on a proper abstract Root_String type. But that wouldn't work in Ada, since it would be incompatible with all of the existing code out there. Probably would have to wait for a follow-on language.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 20 Apr 2021 14:13:30 -0500

> BTW, the very first version of this AI had different types, but the ARG felt that it would just complicate the interface for the sake of abusive "purity".

Unfortunately, that was the first instance that showed the beginning of the end for Ada. If I remember correctly (and I may not ;-), that came from some people who were wedded to the Linux model where nothing is checked (or IMHO, typed). For them, a String is simply a bucket of octets. That prevented putting an encoding of any sort of any type on file names ("it should just work on Linux, that's what people expect"). The rest follows from that.

Those of us who care about strong typing were disgusted, the result essentially does not work on Windows or macOS (which do check the content of file names - as you can see in GNAT compiling units with non-Latin-1 characters in their names), and I don't really expect any recovery from that.

Accessibility Rules and Aliased Parameters

From: Simon Wright <simon@pushface.org>
Subject: GCC 11 bug? lawyer needed
Date: Mon, 03 May 2021 17:08:20 +0100
Newsgroups: comp.lang.ada

This code results in the error shown:

```
1. package Aliased_Tagged_Types is
2.
3.    type T is tagged null record;
4.
5.    function P (Param : aliased T)
      return Boolean
6.      is (False);
7.
8.    function F (Param : T) return
       Boolean
9.      is (Param.P);
              |
   >>> actual for explicitly aliased formal is
too short lived
10.
11. end Aliased_Tagged_Types;
```

The compiler code that results in this error is at sem_ch4.adb:1490, and was introduced for Ada202x accessibility checking reasons.

```
--  Check whether the formal is aliased
-- and if the accessibility level of the
-- actual is deeper than the accessibility
-- level of the enclosing subprogram to
-- which the current return statement
-- applies.

   [...]

   if Is_Explicitly_Aliased (Form)
     and then Is_Entity_Name (Act)
     and then Static_Accessibility_Level
                (Act, Zero_On_Dynamic_Level)
                  > Subprogram_Access_Level
                    (Current_Subprogram)
   then
      Error_Msg_N ("actual for explicitly aliased
      formal is too" & " short lived", Act);
   end if;
```

For those interested, this issue affects Alire.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 4 May 2021 22:54:43 -0500

We spent a lot of time and effort in the ARG talking about this case (see AI12-0402-1). The Ada 2012 RM does indeed say this case is illegal. The reason is that aliased parameters are designed so that one can return part of them in the return object of the function. And a normal parameter is assumed to be local (since its accessibility is unknown) - that means it is too local for an aliased parameter of a function that is used in some non-local way (including being returned from a non-local function).

However, since one cannot return a part of a parameter for a function that returns an elementary type (other than anonymous access returns, which have special rules anyway), we added an exception to the rules for that case in Ada 202x. (We tried a number of more liberal exceptions, but they were complex and had [unlikely] holes.) So the most current rule is that the call of P is legal.

That wasn't decided until the December ARG meeting, so it happened after the GNATPro 21 release (and I expect that the GNAT CE is derived from that version). And I'd guess that in Ada 2012 mode, this check would remain as it is (the change was not made retroactively - not sure why).

From: Adamagica <christ-usch.grein@t-online.de>
Date: Wed, 5 May 2021 03:01:06 -0700

> And a normal parameter is assumed to be local (since its accessibility is unknown) - that means it is too local for an aliased parameter of a function that is used in some non-local way

RM 3.10(9/3): Finally, a formal parameter or generic formal object of a tagged type is defined to be aliased.

RM 6.4.1(6/3): If the formal parameter is an explicitly aliased parameter, the type

of the actual parameter shall be tagged or the actual parameter shall be an aliased view of an object.

Both of these conditions are fulfilled here.

There are many more places about explicitly aliased parameters in the RM. I've read them all. It left me wondering.

I do not see what aliasing a tagged parameter buys. A parameter of a tagged typed is aliased per se, or do I misread the RM.

I'm having big problems trying to understand the RM.

I will try to grock the AI.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Wed, 5 May 2021 09:10:01 -0700

> I will try to grock the AI.

Hm, I'm still confused. Can anyone please come up with some examples that explain what this is all about?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 5 May 2021 19:39:11 -0500

> Can anyone please come up with some examples that explain what this is all about?

See 6.4.1(6/3): there is an accessibility check on the actual parameter of an aliased parameter. This allows an aliased parameter to have the accessibility of the return object of a function, rather than local accessibility. There's a bunch of rules in 3.10.2 that combine to have the right effect.

You see the result in an operation like "Reference" in the containers. If you have:

```
function Foo (A : in out Container;
Idx : in Natural) return access Element;
```

then an implementation of:

```
function Foo (A : in out Container)
return access Element is
begin
   return A.Data(Idx)'Access; -- (1)
end Foo;
```

(1) is illegal, as A has local to Foo accessibility, while the anonymous access has the accessibility of the return object (the point of call), which is necessarily outside of Foo.

You can change (1) to:

```
return A.Data(Idx)'Unchecked_Access;
-- (1)
```

but now you can create a dangling pointer, for instance if Foo is assigned to a library-level access type and the actual for A is not library-level.

But you can change the parameter to "aliased", then the accessibility check is moved to the call site (where it must always succeed for the vast majority of calls). There's no accessibility check at (1) in that case (which could be at best a dynamic check, which is a correctness hazard, and also has an overhead cost). And you still have the safety of not being able to create a dangling pointer.

It is a bit weird that this property is tied to "aliased" parameters. This property came first, and we discussed the syntax to use for a long time. Eventually it was decided to call them "aliased" parameters, but of course that meant it was necessary to generalize the usages.

This special rule does have the downside of being able to fail in some safe cases, like the one noted by the OP. That doesn't happen for procedures, since aliased parameters have no special semantics for procedures. We decided to remove the special semantics for functions for which it is impossible to return a part of the parameter (that is, any elementary-returning function), as that special semantics provides no benefit in such a case (but it does have a cost).

I agree that the original author of that program should not have used "aliased" in the way that they did (they don't need the special semantics), but we realize that some people would prefer to *explicitly* mark things as aliased when they are going to take 'Access (and not worry about the type of the parameter -- after all, it could change). That is, they don't want to depend on the implicit behavior of tagged types -- or perhaps they don't even know about it. Which leads to the problem that occurs here, as "aliased" has slightly different meanings for functions (now just composite functions) and procedures.

Since this is real code that didn't work as expected, it seemed to make sense to reduce the problem with a minor language tweak.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Thu, 6 May 2021 06:07:23 -0700

Thank you, Randy, for the nice explanation. There're still some hazy places, but I begin to see the big picture.

From: Simon Wright <simon@pushface.org>
Date: Thu, 06 May 2021 21:02:54 +0100

The original code, from the Alire project, had (I've edited it slightly)

```
package Holders is new
Ada.Containers.Indefinite_Holders
(Node'Class);

type Tree is new Holders.Holder

  and ...

function Root (This : Tree) return
Node'Class is (This.Constant_Reference);
```

where that Constant_Reference is inherited (eventually) from Ada.Containers.Indefinite_Holders.Holder,

```
function Constant_Reference
  (Container : aliased Holder) return
  Constant_Reference_Type;
pragma Inline (Constant_Reference);
```

Shame it had to be there.

I've just tried splattering 'aliased' wherever the compiler told me it was needed; it's now spreading into other packages. Ugh.

The solution might just be using composition rather than inheritance.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 6 May 2021 22:51:43 +0200

> The solution might just be using composition rather than inheritance.

In my experience mixing handles with target types does not work anyway regardless of the accessibility rules mess.

I tend to use interfaces instead:

```
type Abstract_Node_Interface is interface
...;
```

Then both the handle and the target type implement Abstract_Node_Interface. The target type goes into hiding, the client needs not to see it.

This requires manual delegation in all primitive operations of handles: dereference + call. But in the end it pays off. Especially with trees, because in mutator operations I can check the reference count of the node and choose to clone it (and maybe the subtree) if there are multiple external handles to it.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 6 May 2021 18:59:28 -0500

> function Constant_Reference

> (Container : aliased Holder) return Constant_Reference_Type;

> pragma Inline (Constant_Reference);

Constant_Reference is the case for which these semantics were designed. Hard to avoid it there. ;-)

Note that by returning Node'Class rather than an elementary type, you don't get to use the new rule tweak. Since all tagged types are by-reference (not by copy), the "Root" routine has to return the object that it has, which ultimately is part of Tree. So you actually need "aliased" on Root, since you are (ultimately) returning a part of the formal parameter (and which could become dangling if you pass in an object which is too local).

> I've just tried splattering 'aliased' wherever the compiler told me it was needed; it's now spreading into other packages. Ugh.

I think you need to make a copy of the return object somewhere; the obvious answer is to replace function Constant_Reference with function Element. Of course, if the return object is large enough, that could be expensive. (That doesn't work if you want to write the node, but the use of Constant_Reference doesn't allow that anyway, so in this case it doesn't matter.)

> The solution might just be using composition rather than inheritance.

Yeah, or using handles more as Dmitry says. In any case, it seems like some redesign is necessary.

From: Simon Wright <simon@pushface.org>
Date: Sat, 08 May 2021 11:17:18 +0100

> I think you need to make a copy of the return object somewhere; the obvious answer is to replace function Constant_Reference with function Element.

That appears to be a fine workaround! Thanks!

Stacktraces on Raspberry Pi

From: Björn Lundin <b.f.lundin@gmail.com>
Subject: stacktrace gan on raspberry pi
Date: Mon, 10 May 2021 18:01:50 +0200
Newsgroups: comp.lang.ada

Hi!

I got a raspberry pi 4 - 8 Gb, with

Ubuntu 20.04 LTS on.

I use the GNAT provided by apt -

ubuntu@ubuntu:~/svn/wcs-std/target/message$ gnatls -v

GNATLS 9.3.0

Copyright (C) 1997-2019, Free Software Foundation, Inc.

I seem to get some info out on a crash [...] but addr2line gives

ubuntu@ubuntu:~/svn/wcs-std/target/message$ addr2line -e

./message_utility 0xaaaab8fc2b84 0xaaaab8fc5924 0xaaaab900c364 0xaaaab90024dc 0xaaaab8fbab48 0xaaaab8fbe364 0xaaaab8fb9848 0xffffa687c08c 0xaaaab8fb9898

??:0

[Repeated for every address. —arm]

??:0

Is stacktracing not implemented (which I suspect it is not) or is there another tool to use? (like atos on macos)

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 11 May 2021 13:17:26 +0200

AFAIK and all necessary disclaimers...

You cannot get trace under GNAT ARM, because this is what we recently requested for our GNAT Pro ARM cross compiler. AdaCore confirmed the issue and fixed it for us.

So, in some near future it might arrive at the FSF GNAT.

Proliferation of Reserved Words

From: Jeffrey R. Carter
Subject: Proliferation of Reserved Words
Date: Mon, 31 May 2021 22:51:56 +0200
Newsgroups: comp.lang.ada

Ada 83 (in)famously had 63 reserved words, which was considered a lot at the time (languages like C and Pascal had about half that). Considering only those related to tasking, there were 7:

abort accept do entry select task terminate

Yet many of these have similar/related meanings, and perhaps some overloading would have been a good idea.

entry and accept ... do go hand in hand. One could replace accept with something like an entry body, eliminating 2 reserved words.

An entry is very like a procedure, so one could use procedure instead. It might be necessary to distinguish between a "task procedure" (declared in a task spec) and a "normal procedure" (declared anywhere else). Another reserved word eliminated.

abort/terminate are pretty much the same thing. We could eliminate abort and just use terminate. (One could argue for using end, but given how often "end Name;" appears when not terminating a task, that would be confusing.)

So we're left with select, task, and terminate, less than half as many. I haven't looked in detail at the others, but presumably some reduction is possible there.

Ada 95 added protected and requeue. Some Ada-83 compilers implemented "passive tasks" that were similar to protected objects; formalizing that would have required defining pragma Passive, leaving no need for protected.

There may be a need for requeue, but I've only used it to work around the limitations on what a protected action may do, so I'm skeptical.

ISO/IEC 8652:2007 added synchronized. I think that could have simply reused task.

Ada 12 didn't expand this set of reserved words.

Ada 2X proposes adding parallel. Again, I think reusing task ("task begin" and "task loop") would be fine.

So we will have 11 tasking-related reserved words (unless I've missed some), but we only need select, task, and terminate (and maybe requeue), nearly a factor of 4 difference.

Maybe Ada 3X will add concurrent, and then there won't be any tasking terms that aren't reserved words.

What do others think? Should Ada have made a greater effort at overloading reserved words from the beginning? Should we belatedly object to adding parallel when we have so many choices already? Or is having a large set of reserved words, many of them with similar meanings, a good thing?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 31 May 2021 23:27:49 +0200

I believe that most reserved keywords can be simply unreserved. Actually there is no syntactic necessity except for a few. The rest is kept reserved for the sake of regularity only.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 1 Jun 2021 00:54:31 -0500

At least twice it was proposed that Ada have "keywords", identifiers with special meaning in the syntax but that were not reserved. The last time (and I forget precisely when that was), the ARG had a slight majority in favor of unreserved keywords as well as reserved words. However, it was resoundly rejected at the WG 9 level. At that time, WG 9 still voted by countries, and it turned out that pretty much everyone in favor of unreserved keywords were from North America. Most Europeans were appalled. Of course, that meant a WG 9 vote with 2 in favor and a large number against.

So whenever you think there are too many reserved words in Ada, be assured that it was repeatedly suggested that they not all be reserved, but certain countries would not allow it. At this point, we've given up, since it really would not help much - the majority of words that likely ever be reserved already are (it would most likely matter if a new proposal tried to reserve some commonly used term - "yield" came up some proposals for Ada 202x that didn't go anywhere).

Jeff Carter should note the 8 different uses for "with" in the syntax before he accuses anyone of not reusing reserved words in Ada. It's just the case that it's hard to write something meaningful with the existing reserved words (we almost always try).

"parallel" is an interesting case. In my world view, it is wildly different from a task, because it is *checked*, does not *block* or *synchronize* with another thread (all synchronization is via objects or completion), is automatically created

(in looping constructs) and therefore requires substantial less care than writing a task. There is another world-view where essentially the checking is not worthwhile and ergo must be suppressed, that performance matters to the point at which a compiler isn't allowed to make choices, and essentially requires *more* care than a task. In that second world-view, parallel constructs are either harmful or worthless. But even there, having a keyword makes it a lot easier to avoid them than trying to figure out which libraries to block. :-)

From: Paul Rubin
<no.email@nospam.invalid>
Date: Tue, 01 Jun 2021 00:40:50 -0700

> At least twice it was proposed that Ada have "keywords", identifiers with special meaning in the syntax but that were not reserved.

I remember this as a fundamental decision of PL/I that made PL/I quite hard to parse using the automata-based methods developed not long afterwards. I don't know what consequences that had for PL/I or anything else, if any. But I think it was retrospectively considered a mistake. It's a lot easier to separate parsing and scanning if you can have reserved words.

OTOH I know that C compilers sometimes (usually?) handle typedefs by having the parser tell the scanner to treat the typedef name as keyword-like, after it sees that a typedef has been defined.

From: Jeffrey R. Carter
Date: Tue, 1 Jun 2021 11:51:44 +0200

> At least twice it was proposed that Ada have "keywords", identifiers with special meaning in the syntax but that were not reserved.

Unreserved keywords are one approach, though I'm not aware if they come with any negatives. Then the question becomes which reserved words could become unreserved keywords. (This can be restricted to reserved words related to tasking/concurrency to avoid going through all the reserved words.)

> At that time, WG 9 still voted by countries

Does that imply that the voting has since changed and such a proposal might now be accepted?

> Jeff Carter should note the 8 different uses for "with" in the syntax before he accuses anyone of not reusing reserved words in Ada.

I agree that the ARG has done a good job in reusing reserved words in many cases, "with" being the most obvious. I concentrated on tasking/concurrency reserved words since that seems to be an exception.

From: Robin Vowels
<robin.vowels@gmail.com>
Date: Thu, 3 Jun 2021 01:48:29 -0700

>> At least twice it was proposed that Ada have "keywords", identifiers with special meaning in the syntax but that were not reserved.

> I remember this as a fundamental decision of PL/I that made PL/I quite hard to parse using the automata-based methods developed not long afterwards. I don't know what consequences that had for PL/I or anything else, if any. But I think it was retrospectively considered a mistake.

It was definitely never considered a mistake in PL/I. Not having reserved words means that you do not have to steer clear of using any particular words when you design a program. It also means that a program will continue to compile even when new keywords are introduced into the language. Over the years, new keywords were introduced into PL/I, without invalidating existing programs.

Reserved words are the bane of COBOL.

GNAT in Homebrew

From: Simon Wright
<simon@pushface.org>
Subject: Homebrew, GNAT
Date: Tue, 01 Jun 2021 09:04:41 +0100
Newsgroups: comp.lang.ada

Homebrew (https://brew.sh) is a package manager for macOS.

Since releasing GCC 11.1.0 for macOS (at Sourceforge and now Github[1]), people have been saying what a good idea it would be to have it in Homebrew.

I understand that a binary (pre-built) component for Homebrew is called a "cask". If someone who knows how to build a "cask" wants to do so for GCC+Ada I would help, but for the moment that’s as far as it goes.

From: Simon Wright
<simon@pushface.org>
Date: Tue, 01 Jun 2021 17:10:03 +0100

> Since releasing GCC 11.1.0 for macOS (at Sourceforge and now Github[1]),

[1] https://github.com/simonjwright/building-gcc-macos-native/releases/tag/gcc-11.1.0.1

From: Bill Findlay
<findlaybill@blueyonder.co.uk>
Date: Wed, 02 Jun 2021 00:02:36 +0100

How does that relate to GNAT CE 2021? I see that AdaCore have not released a version for macOS.

From: Simon Wright
<simon@pushface.org>
Date: Wed, 02 Jun 2021 09:51:15 +0100

No; and apparently GNAT CE 2021 is going to be the last CE release for any target.

It is possible to build CE 2021 for macOS (I and another on the GNAT-OSX mailing list are disagreeing somewhat on how to configure for this), but at some point we have to bite the bullet.

What I'm not sure of is gnatprove. If the compiler sources don't match what gnatprove expects you'll get build failures or, at best, runtime failures. And how far could you trust it even if it appeared to work?

Of course, if you need to trust it you'll be happy to pay.

From: Mark Lorenzen
<mark.lorenzen@gmail.com>
Date: Thu, 3 Jun 2021 22:55:56 -0700

> No; and apparently GNAT CE 2021 is going to be the last CE release for any target.

Why do you say that? Can you provide any sources?

From: Simon Wright
<simon@pushface.org>
Date: Fri, 04 Jun 2021 08:38:10 +0100

> Why do you say that? Can you provide any sources?

Of course, I may be macOS-biased here :-)

Remarks at [1],

"We see a majority in favor or recommending GNAT FSF.

"The result is less clear for the removal of GNAT community. The comments along the answers show that people against this are worried about the ease of use. So we are going to work on that aspect."

"We don't expect anyone to build GCC/GNAT themselves, and that is why part of our plan is to help maintainers of OS distribution make good GNAT package."

"AdaCore will continue to provide the SPARK toolset on Linux and Windows. There is no runtime coming with the toolset, so no possible license confusion, it's only an analysis tool!" [[Can we run Linux apps in Docker on a Mac? looks possible, and might I think be an OK solution for gnatprove given the above. M1 macs??]]

and [2],

"Most likely this version of the compiler will be the last in the GNAT Community Edition release chain. In the future, the compiler collected from open source GCC texts can be installed using a batch manager Alire."

[1]

https://www.reddit.com/r/ada/comments/j6oz6i/results_of_the_survey_on_the_future_of_gnat/

[2] https://www.altusintel.com/public-yy39qc/

From: Mark Lorenzen <mark.lorenzen@gmail.com>
Date: Fri, 4 Jun 2021 03:08:16 -0700

Thank you very much. It looks like AdaCore will provide builds of FSF GCC to distro maintainers instead of distributing GNAT as CE. That's fine - as long as I don't have to build GNAT myself from the FSF distro :-)

Ada Lovelace Pint Glass

From: Anatoly Chernyshev <achernyshev@gmail.com>
Subject: Ada Lovelace Pint Glass
Date: Fri, 4 Jun 2021 03:51:38 -0700
Newsgroups: comp.lang.ada

I'm sure many of you will like the idea of having a beer glass dedicated to lady Ada:

https://cognitive-surplus.com/collections/beer-glasses/products/ada-lovelace-pint-glass-1

Web Frontend in Ada 2012

From: Marius Amado-Alves <amado.alves@gmail.com>
Subject: Any chance of programming a web frontend in Ada 2012?
Date: Tue, 8 Jun 2021 01:56:14 -0700
Newsgroups: comp.lang.ada

It seems that currently the only languages web browsers execute reliably are JavaScript and JBC (Java Byte Code), with WASM (Web Assembly) soon to join the group.

Is there a way to program a web frontend in Ada 2012, maybe by translation to one of the above languages?

(Preferably with a binding to the DOM and the BOM.)

(Maybe via LLVM? GNAT already generates LLVM, right?)

Thanks a lot.

From: Jeffrey R. Carter
Date: Tue, 8 Jun 2021 11:21:23 +0200

Have you looked at Gnoga?
https://sourceforge.net/projects/gnoga/

From: Max Reznik <reznik@adacore.com>
Date: Tue, 8 Jun 2021 02:35:26 -0700

Indeed, there is a project to run Ada in the browser using WebAssembly. It's named AdaWebPack[1].

It provides a toolchain based on GNAT LLVM and a customized runtime.

The runtime has some restrictions for now, such as no exception handling due to current state of WebAssembly. The toolchain building could be complicated, so the project provides a Docker image.

The project provides the simplest example (See online: https://www.ada-ru.org/files/wasm/index.html).

This site (in Russian) uses it to provide some construction calculations https://mycalcs.ru/

Also take a look a short blog post: https://blog.adacore.com/android-application-with-ada-and-webassembly

I think, you can reach the author on the Telegram channel https://t.me/ada_lang

[1] https://github.com/godunko/adawebpack

Grom: Marius Amado-Alves <amado.alves@gmail.com>
Date: Tue, 8 Jun 2021 07:45:00 -0700

> Have you looked at Gnoga?
> https://sourceforge.net/projects/gnoga/

Yes. Looks great and reliable. Quick read of the well written user_guide (I must be rainman cause I spotted this typo: Gnoga.Gui.Element.Canvas)

Looks too complicated for my present needs, but definitely a reference to keep. Thanks, Jeff.

From: Marius Amado-Alves <amado.alves@gmail.com>
Date: Tue, 8 Jun 2021 08:01:20 -0700

A number of ideas keep tickling my mind on how to do this. One is using ASIS to translate Ada to JavaScript, a kind of Ada compiler with Javascript as the target language.

From: Max Reznik <reznik@adacore.com>
Date: Tue, 8 Jun 2021 08:22:53 -0700

Speaking about "a single language" for frontend and backend. There was an idea to port Annex E (DSA) to AdaWebPack and use it as a communication channel between a web server written in Ada and WebAssembly client part.

From: Maxim Reznik <reznikmm@gmail.com>
Date: Tue, 8 Jun 2021 09:26:57 -0700

> A number of ideas keep tickling my mind on how to do this. One is using ASIS to translate Ada to JavaScript, a kind of Ada compiler with Javascript as the target language.

I made some progress in this direction, but ASIS4GNAT is abandoned and my project is suspended. The only user I have moved to AdaWebPack :)

The source code of the translator is part of the Matreshka project.

https://forge.ada-ru.org/matreshka/wiki/Web/A2JS

There is GitHub mirror:

https://github.com/reznikmm/matreshka

From: Shark8 <onewingedshark@gmail.com>
Date: Thu, 10 Jun 2021 06:33:55 -0700

> Speaking about "a single language" for frontend and backend. There was an idea to port Annex E (DSA) to AdaWebPack and use it as a communication channel between a web server written in Ada and WebAssembly client part.

I've been advocating this idea [well similar, I'm not a fan of WASM] for years now. Seriously: The DSA has the potential to be the Ada equivalent of being the "killer app" or "killer feature" for getting use.

From: Paul Rubin <no.email@nospam.invalid>
Date: Thu, 10 Jun 2021 17:44:18 -0700

> A number of ideas keep tickling my mind on how to do this. One is using ASIS to translate Ada to JavaScript, a kind of Ada compiler with Javascript as the target language.

It's unclear to me why anyone would want to do this, since it combines the disadvantages of both Javascript (interpreted, non-deterministic execution) and Ada (manual memory management etc.)

If you want to use a typed language that gets translated into Javascript, you might be better off using Purescript (purescript.org) or even something like Agda.

Ada to WASM might make more sense than Ada to Javascript, of course.

Going beyond Ada 2022

From: Stephen Davies <joviangm@gmail.com>
Subject: Adacore Blog - Going Beyond Ada 2022
Date: Tue, 8 Jun 2021 06:28:35 -0700
Newsgroups: comp.lang.ada

The following may be of interest (I was pleased to see fixed lower bounds being considered):

https://blog.adacore.com/going-beyond-ada-2022

From: J-P. Rosen <rosen@adalog.fr>
Date: Wed, 9 Jun 2021 07:11:43 +0200

Of course, everyone is welcome with helping the evolution of Ada. But I remind the community that there is the Ada-Comment list (http://www.ada-auth.org/comment.html) which is open to the public.

I'm afraid that this initiative of AdaCore is another step in trying to control the language.

From: Adamagica <christ-usch.grein@t-online.de>
Date: Wed, 9 Jun 2021 08:10:49 -0700

What troubles me is this statement:

<quote>What happens afterwards

...

Finally, a member of the AdaCore team will give a final decision about the RFC's inclusion in GNAT, and potential submission to the ARG if necessary.</quote>

Sounds like creating dialects. RFCs implemented in GNAT, but not submitted to ARG. What a mess!

Ada is not owned by AdaCore!

From: Paul Rubin
<no.email@nospam.invalid>
Date: Wed, 09 Jun 2021 09:33:10 -0700

It sounded like the RFCs platform is intended to experimentally test proposed new features, which sounds better than standardizing them without testing them first. And hasn't GNAT always has had extensions beyond the ARM Is this anything new?

From: Adamagica
<christ-usch.grein@t-online.de>
Date: Wed, 9 Jun 2021 13:53:10 -0700

Yes, extensions as allowed in the RM - impl-defined pragmas, attributes...

But not extensions with new syntax.

From: Emmanuel Briot
<briot.emmanuel@gmail.com>
Date: Thu, 10 Jun 2021 04:13:26 -0700

I must admit I do not see the grudge here. As I understand it, the goal is indeed to test proposals in practice, before they make it to the standard. There has been a number of evolutions to the language that are not so convenient to use, for instance, or not flexible enough. Having a prototype implementation for people to play with is a nice idea (and what most other languages do in practice). AdaCore does not propose to control the language. As far as I can tell, these prototypes (like early implementation of what is already in Ada 2020) are generally controlled via the -gnatX switch. If you do not use that, then you do not have access to those new proposals either.

This is akin to implementing some pre-processing tool (for instance using ASIS or libadalang) to test those prototypes, except it might be easier to do directly in the compiler for AdaCore. But nothing prevents anyone from writing such a preprocess to test their own proposals.

The language remains controlled by the standard, and although there is a large number of AdaCore employees in the ARG, that's not all of it. The ada-comment list has another purpose than discussing tentative evolution to the language (and email doesn't lend itself too well to that purpose anyway).

From: Andreas Zeurcher
<zuercher_andreas@outlook.com>
Date: Sun, 13 Jun 2021 13:29:44 -0700

> I must admit I do not see the grudge here.

Complaint or concern (for the welfare of Ada) should be the word there. Grudge is a loaded term of long-term hatred, which by definition is absent in this •newly•-arisen topic of 2 competing RFC-esque nonemail commentary/planning/consensus-building forums (ARG's versus AdaCore's). By having 2 competing consensus-building forums, clearly not all the wood can ever truly be behind one arrow in the consensus building, to paraphrase Scott McNealy.

> As I understand it, the goal is indeed to test proposals in practice, before they make it to the standard.

Your wording implies the fundamental danger: If AdaCore productizes a particular design & implementation prior to even submitting an AI to ARG (as "making it to the standard"s body), then the ARG is implicitly compelled to accept or veto, at the wholesale level, the •entirety• of AdaCore's work on this proposed feature

1) when a quite-different alternative might have been wiser and better for Ada or

2) when course-correction at a key point of departure where AdaCore went off-course might have been wiser.

> There has been a number of evolutions to the language that are not so convenient to use, for instance, or not flexible enough. Having a prototype implementation for people to play with is a nice idea

"Having a prototype implementation for people to play with is a nice idea" ••only as long as it remains playing nice & fairly•• at the ARG. As soon as ARG effectively/practically cannot veto & reject some prototype from AdaCore as a partially- or fully-misguided rotten-egg brain-fart, then AdaCore could utilize this technique to try to occlude & preclude all dissenting views in ARG.

> (and what most other languages do in practice).

C++ for example rarely if ever has cases where some core-language feature appeared in GCC or Clang prior to having at least one N-series proposal submitted to the ISO14889 committee (and indeed, not only submitted, but achieving some degree of partial consensus, at least factionally). For example, Clang has automated reference counting (ARC) only in Objective-C-based modes of operation (including in only certain Objective-C-centric situations of Objective-C++), not in C++ proper. Likewise with Microsoft's C++/CLI and C++/CX keeping evolution to the core language separate from the committee-draft-proposal-centric main language—a few nonstandard pragma-esque attributes here & there notwithstanding.

Comparing an ISO/IEC-standardized language's process to, say, Python's is disingenuous, because Python is a language historically with a benevolent dictator-individual and a normative reference-implementation interpreter as 1st-class citizen from which all other Python interpreters or compilers are expected to conform meticulously as 2nd-class citizens. Scala (versus Scala Native) operates much the same way as Python, in this reference-implementation-as-1st-class-citizen-all-others-2nd-class-citizens regard. Certainly what Ada community might fear is a situation where AdaCore's GNAT is the 1st-class citizen reference-implementation to which all other Ada compilers must conform downstream as mere 2nd-class citizens, where Ada would become the Python model and the Scala-ScalaNative model.

> AdaCore does not *propose* to control the language.

(emphasis added)

Not all control schemes in the history of humankind have been publicly announced a priori ahead of time, even by slip-of-the-tongue leaks, let alone full-fledged well-crafted well-publicized proposals. Indeed, some firm control schemes really are pure innocence (not even control schemes at all) at their early stages, only to occur by happenstance as time marches onward (to be realized in historical analysis in retrospect) as quite pernicious after the fact.

> As far as I can tell, these prototypes (like early implementation of what is already in Ada 2020) are generally controlled via the -gnatX switch. If you do not use that, then you do not have access to those new proposals either.

>

> This is akin to implementing some pre-processing tool (for instance using ASIS or libadalang) to test those prototypes, except it might be easier to do direcly in the compiler for AdaCore.

The problem is not implementing industrial-practice prototypes ••of ARG's proposed AIs•• in the GNAT compiler (or any other vendor's compiler). The problem is implementing •nonAIs• (other than pragmas and pragma-esque constructs) in the compiler that then are later harshly utilized as established industrial practice to standardize as close to verbatim from the GNAT implementation as possible, given that no other compiler's design of that feature is as mature. Indeed, ISO/IEC rules strongly favor homologizing •existing•

industrial practice over a standards body pontificating any fresh creativity not yet seen in industrial practice. Homologize is the actual term-of-art there, as utilized throughout ISO, IEC, EU, UN, and other let's-just-get-along international bodies. Any fresh creativity by ARG competing with AdaCore's establish industrial practice goes against the entire concept of homologizing unless ARG (or whichever let's-all-just-get-along homologizing body) can demonstrate that fresh creativity is absolutely necessary, due to insurmountable impracticalities of endorsement of (one of) the existing industrial practice(s) or blending the multiple industrial practices (of which there likely would be none from the other Ada vendors at the time of debate & standardization of the AI).

> But nothing prevents anyone from writing such a preprocess to test their own proposals.

>

> The language remains controlled by the standard, and although there is a large number of AdaCore employees in the ARG, that's not all of it. The ada-comment list has another purpose that discussing tentative evolution to the language (and email doesn't lend itself too well to that purpose anyway).

Then why doesn't AdaCore simply utilize ARG's existing forum of discussion instead of having their own competing forum? It seems that only 2 are needed: ①ARG's comment forum and ②email. It seems that 3 are not needed; it seems that 3's a crowd: ⒶAdaCore's comment forum and ⒷARG's comment forum and Ⓒemailㅤ.

From: John Mccabe
<john@nospam.mccabe.org.uk>
Date: Mon, 14 Jun 2021 10:35:03 -0000

> C++ for example rarely if ever has cases where some core-language feature appeared in GCC or Clang prior to having at least one N-series proposal submitted to the ISO14889 committee

FWIW, though, the C++ committee appear to think it's acceptable to strangle the specification of certain features based on whether or not it makes it hard for a specific compiler to implement.

A few weeks ago I had reason to look into the justification for something in C++ that appeared, to me, to be stupid and illogical, and the reasoning was because "clang does something this way and, if we took the sensible approach for this feature, it would mean clang would have to massively change".

So now the C++ world is saddled with a specification that's compromised by specific implementations.

I've been trying to find the discussion I had about this with some of my colleagues at work; if I do, I'll let you know what it is!

However, this is, basically, a potential risk with the AdaCore RFC approach; if they forward the feature to the ARG and the ARG comes back with "well, nice, but it would be better if it did this", and AdaCore say "but that would be too hard for us now", then what happens?

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Mon, 14 Jun 2021 17:50:26 -0500

> AdaCore say "but that would be too hard for us now", then what happens?

This is a double-edged sword, of course; if something is hard to implement (even if better), it might never get adopted at all (look at Algol 68 for a historical example of a committee ignoring practical considerations).

And this sort of thing has occurred as far back as Ada 9x: various Ada 9x proposals were withdrawn because of opposition from implementers (especially DEC, which never actually built an Ada 95 compiler). Perhaps it would have been better if those proposals had gone forward, but that's hard to say. It's also possible that those proposals would have prevented construction of some Ada 95 compilers.

My point is that there needs to be a balance; one should not let one implementer or one group dictate everything, but one cannot ignore implementers either. (A corollary to that: the implementers should not ignore the ARG, either! That happened to some degree with Ada 202x.)

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.

The COVID-19 pandemic had a catastrophic impact on conferences world-wide. Where available, the status of events is indicated with the following markers: "(v)" = event is held online, "(h)"= event is held in a hybrid form (i.e. partially online).

2021

July 07-09 (v) — 33rd **Euromicro Conference on Real-Time Systems** (ECRTS'2021), Modena, Italy. Topics include: all aspects of timing requirements in computer systems; elements of time-sensitive computer systems, such as operating systems, hypervisors, middlewares and frameworks, programming languages and compilers, runtime environments, ...; classic worst-case execution time (WCET) analysis; formal methods for the verification and validation of real-time systems; the interplay of timing predictability and other non-functional qualities such as reliability, security, quality of control, scalability, ...; foundational scheduling and predictability questions, such as schedulability analysis, locking and non-blocking synchronization protocols, computational complexity, ...; etc.

☺ July 12-13 (v) — 14th **International Symposium on High-Level Parallel Programming and applications** (HLPP'2021), Cluj-Napoca, Romania. Topics include: high-level parallel programming and tools; high-level parallelism in programming languages; efficient code generation, auto-tuning and optimization for parallel and distributed programs; model-driven software engineering for parallel and distributed systems; applications of parallel and distributed systems using high-level languages and tools; teaching experience with high-level tools and methods for parallel and distributed computing; etc.

☺ July 12-16 (v) — 35th **European Conference on Object-Oriented Programming** (ECOOP'2021), Aarhus, Denmark. Topics include: design, implementation, optimization, analysis, testing, verification, and theory of programs, programming languages, and programming environments.

- ☺ July 13 (v) — 23rd **Workshop on Formal Techniques for J(ust-about-any) Programs** (FTfJP'2021). Topics include: current and novel techniques for formal reasoning about programs, 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.

July 12-16 (v) — 45th **Annual IEEE Conference on Computers, Software and Applications** (COMPSAC'2021), Madrid, Spain.

- July 12-16 — 1st IEEE **International Workshop on Software Engineering for Industrial Cyber-Physical Systems** (SE4ICPS'2021). Topics include: middleware design for industrial IoT/CPS; software design theory for IoT/CPS; formal Methods for IoT/CPS; safety-critical cyber-physical software systems; software quality attributes of IoT/CPS; fault-tolerant IoT/CPS; testing, validation, verification, simulation, and visualization of IoT/CPS; IoT/CPS engineering Methods and Tools; etc.

☺ August 18-20 (v) — 27th IEEE **International Conference on Embedded Real-Time Computing Systems and Applications** (RTCSA'2021), Internet. Topics include: real-time scheduling, timing analysis, formal methods for temporal guarantees, programming languages and run-time systems, middleware systems, applications and case studies of IoT and CPS, cyber-physical co-design, medical CPS, multi-core embedded systems, fault tolerance and security, etc. Deadline for submissions: July 1, 2021 (WiP abstracts).

August 23-27 (v)
29th ACM **Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering** (ESEC/FSE'2021), Athens, Greece.

☺ August 24-26 (v)
25th **International Conference on Formal Methods for Industrial Critical Systems** (FMICS'2021), Paris, France. Co-located with CONCUR'2021 and FORMATS'2021. Topics include: case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; methods, techniques and tools to support automated analysis, certification, debugging, descriptions, learning, optimisation and transformation of complex, distributed, real-time, embedded, mobile and autonomous systems; verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability (e.g., scalability and usability issues); impact of the adoption of formal methods on the development process and associated costs; application of formal methods in standardisation and industrial forums.

☺ Aug 30 - Sep 03 (v)
27th **International European Conference on Parallel and Distributed Computing** (Euro-Par'2021), Lisbon, Portugal. Topics include: all flavors of parallel and distributed processing, such as compilers, tools and environments, scheduling and load balancing, theory and algorithms for parallel and distributed processing, parallel and distributed programming, interfaces, and languages, multicore and manycore parallelism, etc.

Aug 30 – Sep 03 (v)
24th **Ibero-American Conference on Software Engineering** (CIbSE'2021), Costa Rica. Event includes Software Engineering Track (SET) and Experimental Software Engineering Track (ESELAW).

September 02-05 (v)
16th **Federated Conference on Computer Science and Information Systems** (FedCSIS'2021), Sofia, Bulgaria. Event includes: Scalable Computing (12th Workshop WSC'21), Cyber Security, Privacy and Trust (2nd International Forum NEMESIS'21), Cyber-Physical Systems (8th Workshop IWCPS-8), Software Engineering (41th IEEE Workshop SEW-41), Advances in Programming Languages (8th Workshop WAPL'21), Recent Advances in Information Technology (7th Doctoral Symposium DS-RAIT'21), etc.

September 06-10 (h)
18th **International Colloquium on Theoretical Aspects of Computing** (ICTAC'2021), Nur-Sultan, Kazakhstan. Topics include: semantics of programming languages; theories of concurrency; theories of distributed computing; models of objects and components; timed, hybrid, embedded and cyber-physical systems; static analysis; software verification; software testing; model checking and theorem proving; etc.

September 07-10 (h)
40th **International Conference on Computer Safety, Reliability and Security** (SafeComp'2021), York, UK. Deadline for early registration: July 16, 2021.

September 08-11 (v)
14th **International Conference on the Quality of Information and Communications Technology** (QUATIC'2021), Faro, Portugal. Topics include: all quality aspects in ICT systems engineering and management; quality in ICT process, product and applications domains; practical studies; etc. Tracks on ICT verification and validation, safety, security and privacy, model-driven engineering, quality in cyber-physical systems, software evolution, evidence-based software quality engineering, software quality education and training, etc.

September 21-23 (h)
20th **International Conference on Intelligent Software Methodologies, Tools and Techniques** (SOMET'2021), Cancun, Mexico. Topics include: state-of-art and new trends on software methodologies, tools and techniques; software methodologies, and tools for robust, reliable, non-fragile software design; software developments techniques and legacy systems; software evolution techniques; agile software and lean methods; formal methods for software design; software maintenance; software security tools and techniques; formal techniques for software representation, software testing and validation; software reliability; Model Driven Development (DVD), code centric to model centric software engineering; etc.

October 3-7 (h)
16th **International Conference on Software Engineering Advances** (ICSEA'2021), Barcelona, Spain. Topics include: trends and achievements; advanced fundamentals, mechanisms, or design tools for developing software; software security, privacy, safeness; advances in software testing; specialized software 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: July 5, 2021.

October 8-15 (v)
Embedded Systems Week 2021 (ESWEEK'2021). Shanghai, China. The venues for ESWEEK 2020 and 2021 were swapped. ESWEEK 2020 was to be held in Hamburg, Germany from September 20-25,

2020. ESWEEK 2021 would be held in Shanghai, China from October 10-15, 2021, but then moved to a virtual event format. Includes CASES'2021 (International Conference on Compilers, Architectures, and Synthesis for Embedded Systems), CODES+ISSS'2021 (International Conference on Hardware/Software Codesign and System Synthesis), EMSOFT'2021 (International Conference on Embedded Software).

Oct 10-15 (v) — ACM SIGBED **International Conference on Embedded Software** (EMSOFT'2021). Topics include: the science, engineering, and technology of embedded software development; research in the design and analysis of software that interacts with physical processes; results on cyber-physical systems, which integrate computation, networking, and physical dynamics.

Oct 10-15 (v) — **International Conference on Hardware/Software Codesign and System Synthesis** (CODES+ISSS'2021). Topics include: system-level design, hardware/software co-design, modeling, analysis, and implementation of modern Embedded Systems, Cyber-Physical Systems, and Internet-of-Things, from system-level specification and optimization to system synthesis of multi-processor hardware/software implementations.

Oct10-15 (v) — **International Conference on Compilers, Architecture, and Synthesis for Embedded Systems** (CASES'2021). Topics include: latest advances in compilers and architectures for high-performance, low-power, and domain-specific embedded systems; compilers for embedded systems: multi- and many-core processors, GPU architectures, reconfigurable computing including FPGAs and CGRAs, security, reliability, and predictability (secure architectures, hardware security, and compilation for software security; architecture and compiler techniques for reliability and aging; modeling, design, analysis, and optimization for timing and predictability; validation, verification, testing & debugging of embedded software); etc.

October 11-14 (h) — 21st **International Conference on Runtime Verification** (RV'2021), Los Angeles, California, USA. Topics include: monitoring and analysis of runtime behaviour of software and hardware systems. Application areas include cyber-physical systems, safety/mission critical systems, enterprise and systems software, cloud systems, autonomous and reactive control systems, health management and diagnosis systems, and system security and privacy, among others.

October 11-15 (h) — 15th ACM/IEEE **International Symposium on Empirical Software Engineering and Measurement** (ESEM'2021), Bari, Italy. ESEM'2020 was postponed from 8-9 October 2020 to 2021. Deadline for submissions: August 9, 2021 (Journal-First papers, industry talks).

☺ October 17-22 (h) — ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2021), Chicago, Illinois, USA. Topics include: all aspects of software construction and delivery, at the intersection of programming, languages, and software engineering. Deadline for submissions: July 5, 2021 (GPCD - 20th International Conference on Generative Programming: Concepts & Experiences), July 16, 2021 (student research competition), July 16, 2021 (SPLASH-E), August 6, 2021 (workshop papers), August 15, 2021 (SPLASH posters). Deadline for early registration: September 17, 2021.

Oct 17-19 — 14th ACM SIGPLAN **International Conference on Software Language Engineering** (SLE'2021). Topics include: areas ranging from theoretical and conceptual contributions, to tools, techniques, and frameworks in the domain of software language engineering; software language engineering rather than engineering a specific software language; software language design and implementation; software language validation; software language integration and composition; software language maintenance (software language reuse, language evolution, language families and variability); 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: July 5, 2021 (abstracts), July 9, 2021 (papers), September 15, 2021 (artifacts).

October 17-22 — 28th **Static Analysis Symposium** (SAS'2021), Chicago, Illinois, USA. In conjunction with SPLASH'2021. Topics include: static analysis as fundamental tool for program verification, bug detection, compiler optimization, program understanding, and software maintenance.

October 18-22 (v) — 19th **International Symposium on Automated Technology for Verification and Analysis** (ATVA'2021), Gold Coast, Australia. Topics include: theoretical and practical aspects of automated analysis, synthesis, and verification of hardware, software, and machine learning (ML) systems; program analysis and software verification; analytical techniques for safety, security, and dependability; testing and runtime analysis based on verification technology; analysis and verification of parallel and concurrent systems; verification in industrial practice; applications and case studies; automated tool support; etc.

November 15-19 (v) — 36th IEEE/ACM **International Conference on Automated Software Engineering** (ASE'2021), Melbourne, Australia. Topics include: foundations, techniques, and tools for automating the analysis, design, implementation, testing, and maintenance of large software systems; testing, verification, and validation; software analysis; empirical software engineering; maintenance and evolution; software security and trust; program comprehension; software architecture and design; reverse engineering and re-engineering; model-driven development; specification languages; software product line engineering; etc. Deadline for submissions: August 20, 2021 (workshop papers).

November 20-26 (v) — 24th **International Symposium on Formal Methods** (FM'2021), Beijing, China. Topics include: formal methods in a wide range of domains including software, computer-based systems, systems-of-systems, cyber-physical systems, security, human-computer interaction, manufacturing, sustainability, energy, transport, smart cities, and healthcare; formal methods in practice (industrial applications of formal methods, experience with formal methods in industry, tool usage reports, experiments with challenge problems); tools for formal methods (advances in automated verification, model checking, and testing with formal methods, tools integration, environments for formal methods, and experimental validation of tools); formal methods in software and systems engineering (development processes with formal methods, usage guidelines for formal methods, and method integration); etc.

November 22-23 (v) — 15th **International Conference on Verification and Evaluation of Computer and Communication Systems** (VECoS'2021), Beijing, China. Topics include: formal verification and evaluation approaches, methods and techniques, especially those developed for concurrent and distributed hardware/software systems; abstraction techniques; compositional verification; correct-by-construction design; rigorous system design; model-checking; performance and robustness evaluation; QoS evaluation, planning and deployment; dependability assessment techniques; RAMS (Reliability-Availability-Maintainability-Safety) assessment; model-based security assessment; verification & validation of IoT and of safety-critical systems; assessment for real-time systems; worst-case execution time analysis; etc. Application areas include: communication protocols, cyber-physical systems, high-performance computing, internet of things, logistics systems, mixed criticality systems, programming languages, real-time and embedded operating systems, telecommunication systems, etc. Deadline for submissions: July 20, 2021 (papers).

November 25-26 — 22nd **International Conference on Product-Focused Software Process Improvement** (PROFES'2021), Turin, Italy. Topics include: experiences, ideas, innovations, as well as concerns related to professional software development and process improvement driven by product and service quality needs. Deadline for submissions: July 5, 2021 (full research paper abstracts), July 12, 2021 (full research papers), July 16, 2021 (short papers, industry papers), August 9, 2021 (Journal-First papers).

December 06-09 (v) — 28th **Asia-Pacific Software Engineering Conference** (APSEC'2021), Taiwan. Topics include: agile methodologies; component-based software engineering; cyber-physical systems and Internet of Things; debugging and fault localization; embedded real-time systems; formal methods; middleware, frameworks, and APIs; model-driven and domain-specific engineering; open source development; parallel, distributed, and concurrent systems; programming languages and systems; refactoring; reverse engineering; security, reliability, and privacy; software architecture, modelling and design; software comprehension and traceability; software engineering education; software engineering tools and environments; software maintenance and evolution; software product-line engineering; software reuse; software repository mining; testing, verification, and validation; etc. Deadline for submissions: July 1, 2021 (technical/SEIP research paper abstracts), July 8, 2021 (technical/SEIP research papers), July 15, 2021 (workshops), August 19, 2021 (ERA - Early Research Achievements papers), October 7, 2021 (poster papers).

December 06-10 (v) — 24th **Brazilian Symposium on Formal Methods** (SBMF'2021), Campina Grande, PB, Brazil. Topics include: development, dissemination, and use of formal methods for the construction of high-quality computational systems; applications of formal methods to software design, development, code generation, testing, maintenance, evolution, reuse, ...; specification and modelling languages (logic and semantics for specification or/and programming languages; formal methods for timed, real-time, hybrid, or/and safety-critical systems; formal methods for cyber-physical systems; ...); theoretical foundations (type systems models of concurrency, security, ...); verification and validation (abstraction, modularization or/and refinement techniques, static analysis, model checking, theorem proving, software certification, correctness by construction); experience reports on teaching formal methods, on industrial application of formal methods. Deadline for submissions: July 23, 2021 (abstracts), July 30, 2021 (full papers).

☺ Dec 07-10 (h) — 42nd IEEE **Real-Time Systems Symposium** (RTSS'2021), Dortmund, Germany. RTSS'2021 was moved from Taipei, Taiwan, to Dortmund, Germany. Topics include: addressing some form of real-time requirements such as deadlines, response times or delays/latency; real-time system track (middleware, compilers, tools, scheduling, QoS support, testing and debugging, design and verification, modeling, WCET analysis, performance analysis, fault tolerance, security, system experimentation and deployment experiences, ...); design and application track (cyber-physical systems design methods, tools chains, security and privacy, performance analysis, robustness and safety, analysis techniques and tools, ...; architecture description languages and tools; Internet of Things (IoT) aspects of scalability, interoperability, reliability, security, middleware and programming abstractions, protocols, modelling, analysis and performance evaluation, ...); etc. Deadline for submissions: September 3, 2021 (brief presentations), September 7, 2021 (*RTSS@Work demos), October 1, 2021 (TCRTS Test of Time Award nominations).

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

2022

January 17-19 — 17th **International Conference on High Performance and Embedded Architecture and Compilation** (HiPEAC'2022), Budapest, Hungary. Topics include: computer architecture, programming models, compilers and operating systems for embedded and general-purpose systems.

April 02-07 — 25th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2022), Munich, Germany. Events include: ESOP (European Symposium on Programming), FASE (Fundamental Approaches to Software Engineering), FoSSaCS (Foundations of Software Science and Computation Structures), TACAS (Tools and Algorithms for the Construction and Analysis of Systems). Deadline for submissions: August 30, 2020 (nominations EAPLS PhD Award), October 14, 2021 (papers).

♦ June 14-17 (h) — 26th **Ada-Europe International Conference on Reliable Software Technologies** (AEiC 2022 aka Ada-Europe 2022), Ghent, Belgium. Sponsored by Ada-Europe.

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

We hope you enjoyed AEiC 2021

It surely was an interesting experience: the virtual platform took some getting used to, but it definitely allowed a fair amount of contact and interaction

Thanks to the progress with the vaccination campaign, AEiC 2022 will return to the **in-presence** style

We will however provide solutions for remote participation for those unable or unwilling to travel

The in-presence program will be rich with social, cultural and touristic opportunities

AEiC 2022 will take place in the week of **14-17 June**, in **Ghent**, **Belgium**

Flanders, very near the geographical centre of the town, 30 minutes from Brussels

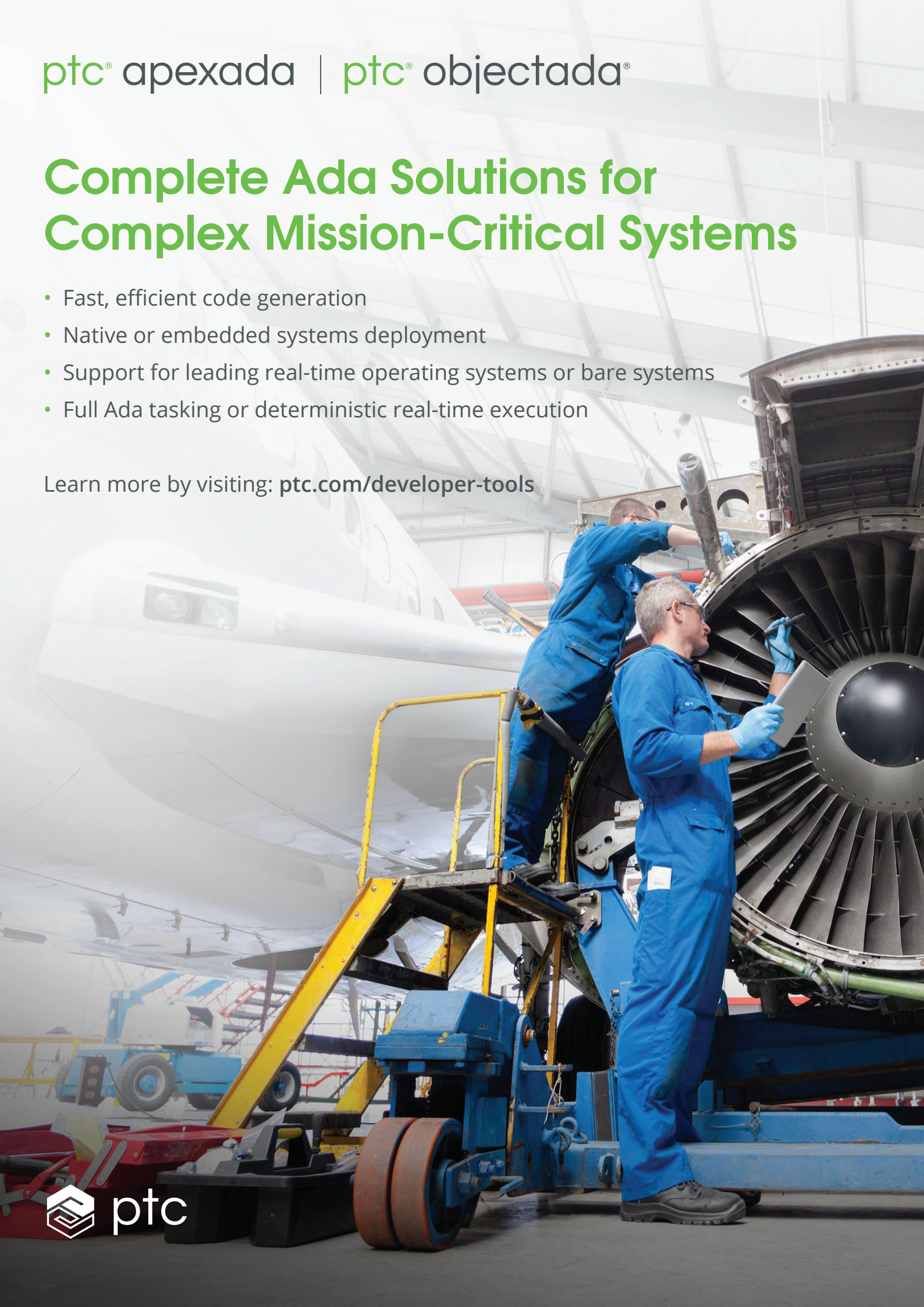
Place of a vibrant university (founded in 1817), but also with a very traditional look

ptc® apexada | ptc® objectada®

Complete Ada Solutions for Complex Mission-Critical Systems

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

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



ptc

How windows size and number can influence the schedulability of hierarchically-scheduled time-partitioned distributed real-time systems

A. Amurrio, E. Azketa

Ikerlan Technology Research Centre (BRTA Member), Arrasate-Mondragón (Spain); email: {aamurrio, eazketa}@ikerlan.es

M. Aldea, J.J. Gutiérrez

Software Engineering and Real-Time group, University of Cantabria. Santander (Spain); email: {aldeam, gutierjj}@unican.es

Abstract

Partitioning techniques are implemented in the development of safety-critical applications to ensure isolation among components. An adequate scheduling of the execution of such partitions is a key challenge so that applications meet the hard deadlines imposed. In this work, we study the effect of different partition window configuration parameters, with the aim of analyzing their impact in the worst-case response times of system tasks. This is the first step in the development of an algorithm for optimizing partition windows in hierarchically-scheduled time partitioned distributed systems.

Keywords: Time-partitioning, Partition-scheduling, Schedulability-analysis, Safety-critical applications.

1 Introduction

Although scheduling real-time systems has been addressed for many years, novel design, architectural and execution paradigms enforce real-time researchers to continuously develop new strategies to guarantee that deadlines imposed in software are met even in the worst-case scenario. Partitions are strictly independent execution environments protected from each other, and they are used when application components have different criticality levels; it is crucial to avoid that low criticality components jeopardize the execution of high criticality ones. Recent works [1] [2] remark the interest on virtualization/partitioning techniques at many industrial domains, such as automotive or railway, as key issues to be addressed when developing safety-critical applications, and in [3] the will of train manufacturers to re-factor their applications designs is shown, in order to allow the execution of applications with different criticality levels.

In this work, we focus on a railway signalling application. Up to now, this application has been executed in a cyclic scheduler where all functions are executed even if they do not have any useful work to do. With the aim of substituting this architecture by a novel partitioned execution environment, in [4] a schedulability analysis technique was presented, and based on this, we can propose a partition scheduling optimization algorithm. A more detailed description of this application can be found in [4], and from now on we will construct a simplified (yet representative) system that captures the most relevant features of such safety-critical applications. The experiments conducted in this work will serve as a basis and guide for the development of an optimization algorithm that shall produce schedulable solutions for time-partitioned distributed real-time systems.

The rest of the paper is organized as follows. In Section 2 the system model addressed in this work is described, including an overview of the response time analysis technique. In Section 3 the experiments conducted are presented, and in Section 4 we draw the conclusions and the future works.

2 System Model

In this work we follow a system model compliant with MAST (Modeling and Analysis Suite Tool for Real-Time Applications) [5] [6], which is a GPL open source model and also a set of scheduling, analysis and simulation tools developed by the University of Cantabria.

2.1 Logical Architecture

The main element of this model is the distributed *end-to-end flow* (hereafter e2e flow) as the one depicted in Figure 1, which consists of a sequence of activities with precedence relations executed in response of a periodic or sporadic workload event (e_{in}), with a minimum inter-arrival time (T_i). The main component of an e2e flow is the event handler called *step*, which represents an operation being executed by a schedulable resource (a task or a message) in a processing resource (a computer or a network). Each step is activated by an input event, and after its execution it generates an output event. *Fork* and *Join* event handlers are also allowed, they do not have runtime effects and enable modeling multipath e2e flows. The j-th step in a Γ_i e2e flow is denoted as τ_{ij}, and it has a worst-case and a best-case execution time, C_{ij} and C_{ij}^b respectively, and a deadline D_{ij} relative to the workload event. Each step represents a utilization of the processing resource of $U_{ij} = C_{ij}/T_i$.

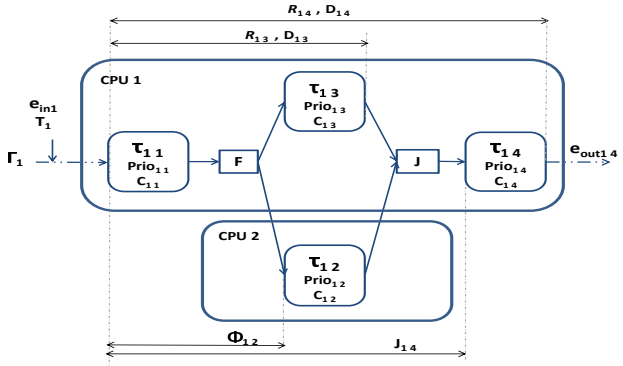


Figure 1: Distributed multipath e2e flow

2.2 Partitioning model

In this work, hierarchically scheduled and time-partitioned systems are addressed. A timetable driven scheduling policy is considered as primary scheduler in every processor, where temporal partitions are scheduled in a cyclic manner. A temporal partition P_x is composed of one or more partition windows Win_{xk} within a periodic Major Frame (MAF). Partition windows start at a time S_{xk} relative to the start of the MAF, and their length is L_{xk}. Hence, partition windows within a temporal partition are defined as follows: Win_{xk} = { S_{xk} , L_{xk} }. Inside each partition, the secondary scheduler is based on preemptive fixed priorities, where $Prio_{ij}$ is the priority of step τ_{ij}, and where the highest number represents the highest priority. The partition utilization of P_x is the sum of the utilization of all the steps contained in that partition: $U_{P_x} = \sum_{\forall \tau_{ij} \in P_x} U_{ij}$. We also define the term Available Utilization (AU_{P_x}) as the processing time allocated to the partition P_x in each processor, which is: $AU_{P_x} = \sum_{\forall Win_{xk} \in P_x} L_{xk}/MAF$.

We are taking into account the overheads provoked by context switches at the primary scheduler. This overhead is the time CS that CPUs need to load a partition context at the beginning of a partition window and saving it after execution terminates. In other words, it can be understood as a non-available CPU time whenever a partition window executes. For response time analysis purpose, this effect can be modelled by gathering this unavailable time at the beginning of every partition window and substracting this amount to the available CPU-time for that window, as shown in the example of Figure 2, where $MAF = 40ms$. In this example a time partition is composed of two partition windows, and the effect of substracting the time for switching the context at each window provokes that the effective partition's (P'_i) length is 2 ms lower than the original P_i. Therefore, the effective partition window is defined as follows: Win'_{ik} = { $S_{ik} + CS$, $L_{ik} - CS$}.

2.3 Response time analysis

The analysis of a given partition is performed independently: the rest of time-partitions, as well as the unused CPU time within the MAF, are modeled as a single high priority e2e flow, called *unavailability flow*, which has to be analyzed along with the steps hosted in the partition under analysis. To do so, we use the offset-based analysis techniques [7]

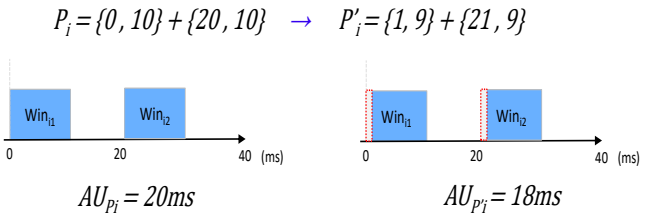


Figure 2: Partition and effective partition with $CS = 1ms$

[8], which was extended in [4] to support the analysis of multipath flows like those addressed in our use-case. Readers are encouraged to read the aforementioned references for a deeper understanding of the schedulability analysis of these systems.

3 Study of the influence of partition windows in schedulability

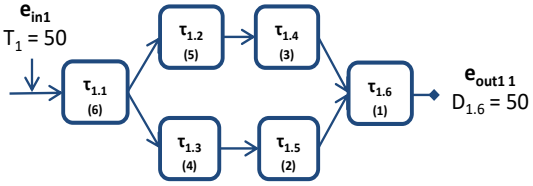


Figure 3: Guiding application example

To perform the proposed study, we will evaluate different partition scheduling schemes. Thus, we are going to construct a simple application example, which gathers the most relevant features that characterize our motivating railway use-case. This example is composed of a single multipath e2e flow activated periodically every 50 ms which is composed of six steps (τ_{11} to τ_{16}), as shown in Figure 3. The flow is mapped within a single partition (P_1), and we assume, for the sake of simplicity, the the execution time of each step is fixed and equals 2 ms. The priority of each step is shown between brackets. When referring to the schedulability of the application, it regards to the worst-case response time of the 6-th step (R_{16}) in comparison with its deadline. The MAF considered for the whole experiment set is 50 ms.

A very common early-design decision regards to the CPU time allocated for the execution of each partition, i.e. AU_{P_1} in our model. Depending on this time, response times may vary significantly as shown in Figure 4. Even if this effect may result obvious, it gives us an idea about the effect that not having all the processor time dedicated for the execution of applications produces on the worst-case response times calculated by the analisis technique. The longer the gaps between partition windows within the MAF are (where P_1 is not allowed to be executed), the higher is the worst-case response time. In this example the partition utilization U_{P_1} represents 24% of the CPU time, and worst-case response times vary from 580 ms to 12 ms when the available CPU goes from the initial utilization of 24% to the 100%.

Once the available time has been fixed, the next design decision to take would be to distribute this given time along

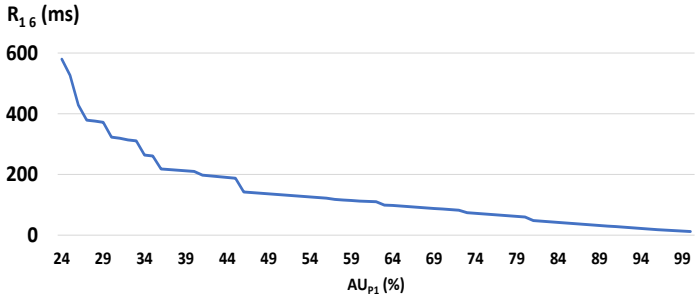


Figure 4: Worst-case response time as a function of P1's available CPU time (in %)

the MAF. At a first approach, we will consider a harmonic distribution of partition windows, although this might be subject for optimization when more partitions conform the MAF. Figure 5 shows the worst-case response times obtained when the number of partition windows varies from 1 to 100, for three different values of AU_{P_1}. As can be seen, increasing the number of windows produces in fact a reduction in the unavailable gaps in the MAF, making a remarkable reduction of the response times obtained.

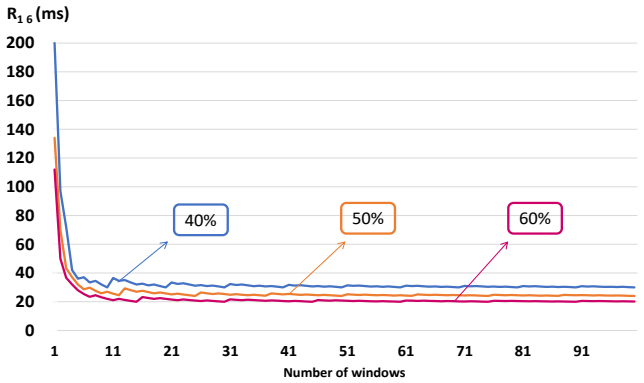
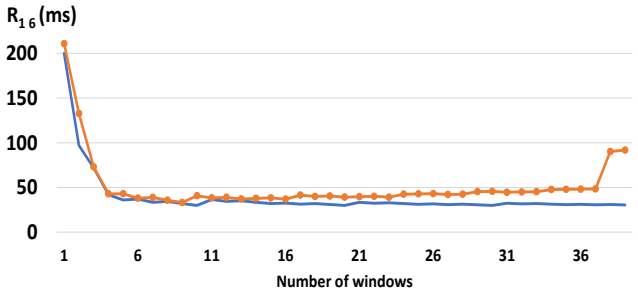


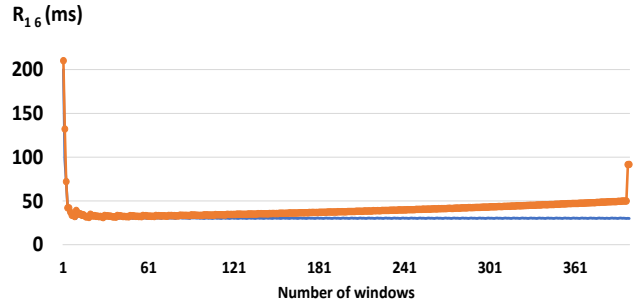
Figure 5: Evolution of worst-case response times when the number of windows is increased, for different % of AU_{P_1}

Experiments conducted until now have not considered the effects of the context switch overheads that are present when a partition is activated, and which have been modeled in the previous section. In general, context switch overheads depend on the operating system/hypervisor where applications are executed. In [9] the measured context switch overheads are 17μs, and in [10] they are 27μs. Therefore, in our experiments we will consider a value of $CS = 20\mu$s and also a higher order of magnitude representing a slower processor, i.e. $CS = 200\mu$s. The maximum CPU time that can be dedicated for context switch overheads is the difference between the available partition utilization (AU_{P_1}) and the partition utilization (U_{P_1}). With this, we determine the limit of the number of partition windows that can be set without overloading the partition as follows:

$$NW_{P_1} = \lfloor (AU_{P_1} - U_{P_1}) * \frac{MAF}{CS} \rfloor \qquad (1)$$



(a) $CS = 200\mu$s



(b) $CS = 20\mu$s

Figure 6: Worst-case response time as a function of the number of partition windows - $AU_{P_1} = 40\%$

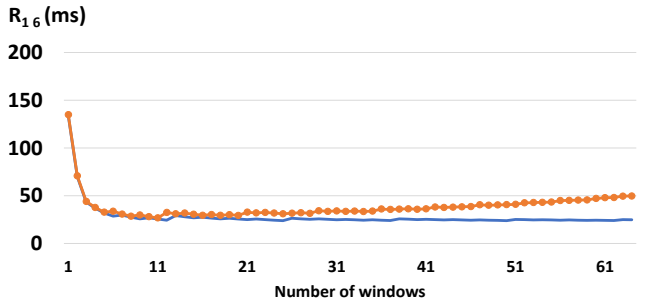
In Figures 6 to 8 the worst-case response times obtained in different scheduling schemes are shown. For different AU_{P_1} values we calculate the response times when increasing the partition window number. Qualitatively, worst-case response times vary in the same way regardless the CPU availability, i.e. the maximum response times are obtained when P_1 is scheduled in a single window, and as the number of windows increases response times reduce fast, up to a point. After that point, notice that in the ideal scenario ($CS = 0$, blue plot) the response-time curve remains constant, while if $CS > 0$ (orange plot) the curve increases again.

4 Conclusions and future work

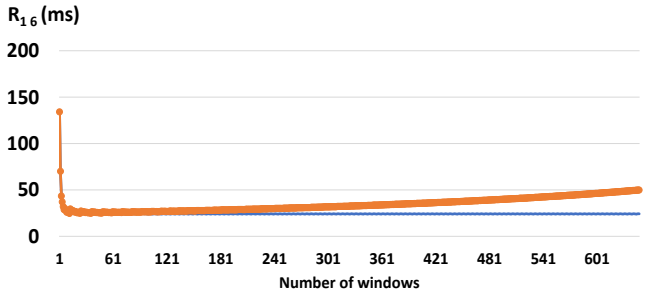
In this work we have presented a study of the behaviour that time-partitioned real-time systems exhibit when worst-case analysis is applied on them, including the effect of context switch overheads on the primary scheduler. The knowledge of this performance will be exploited for developing a partition-scheduling optimization algorithm.

We have observed that increasing the number of partition windows has a positive effect on the reduction of response times, which is outweighed by the negative effect of context switch overheads. Finding the window configuration that gets the turning point on response times is essential for a partition window-optimization algorithm.

When the partition utilization is very low in comparison to the available utilization, and also when context switch overheads are very low, the range of windows to explore ($1..NW_{P_x}$) is very high. We have noticed that analyzing a very big window number (> 500) has a negative impact in the execution time of the analysis tool. That is why the strategy we shall follow is to explore first the minimum number of windows and increasing them until the turning point is found.

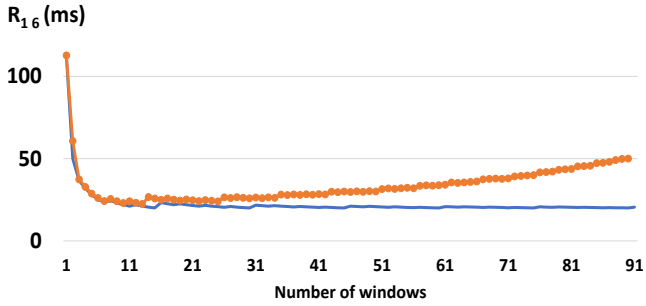


(a) $CS = 200\mu$s

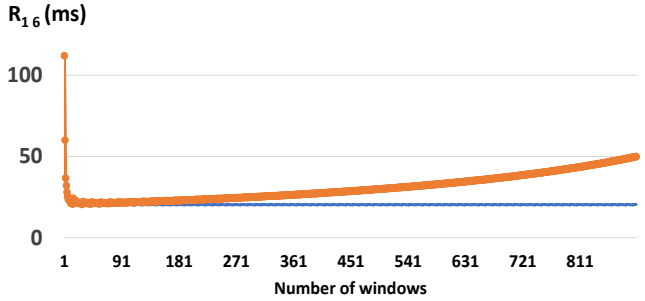


(b) $CS = 20\mu$s

Figure 7: Worst-case response time as a function of the number of partition windows - $AU_{P_1} = 50\%$



(a) $CS = 200\mu$s



(b) $CS = 20\mu$s

Figure 8: Worst-case response time as a function of the number of partition windows - $AU_{P_1} = 60\%$

Acknowledgements

Work supported in part by "Doctorados Industriales 2018" program from the University of Cantabria and by the Spanish Government and FEDER funds (AEI/FEDER, UE) under Grant TIN2017-86520-C3-3-R (PRECON-I4).

References

[1] C. Donnarumma, A. Biondi, F. De Rosa, and S. Di Carlo, "Integrating online safety-related memory tests in multicore real-time systems," In Proceedings of the 41st IEEE Real-Time Systems Symposium (RTSS), 2020.

[2] Z. Jiang, S. Zhao, P. Dong, D. Yang, R. Wei, N. Guan, and N. Audsley, "Re-thinking mixed-criticality architecture for automotive industry," in *Proceedings of the IEEE 38th International Conference on Computer Design (ICCD)*, pp. 510–517, 2020.

[3] H. Fang and R. Obermaisser, "Execution environment for mixed-criticality train applications based on an integrated architecture," in *2017 International Conference on Promising Electronic Technologies (ICPET)*, pp. 1–7, IEEE, 2017.

[4] A. Amurrio, E. Azketa, J. J. Gutierrez, M. Aldea, and M. G. Harbour, "Response-time analysis of multipath flows in hierarchically-scheduled time-partitioned distributed real-time systems," *IEEE Access*, vol. 8, pp. 196700–196711, 2020.

[5] M. González Harbour, J. J. Gutiérrez, J. C. Palencia, and J. M. Drake, "Mast: Modeling and analysis suite for real time applications," in *in Proceedings of the 13th Euromicro Conference on Real-Time Systems*, pp. 125–134, IEEE, 2001.

[6] M. G. Harbour, J. J. Gutiérrez, J. M. Drake, P. López, and J. C. Palencia, "Modeling distributed real-time systems with mast 2," *Journal of Systems Architecture*, vol. 59, no. 6, pp. 331–340, 2013.

[7] J. C. Palencia and M. González Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No. 98CB36279)*, pp. 26–37, IEEE, 1998.

[8] J. C. Palencia, M. González Harbour, J. J. Gutiérrez, and J. M. Rivas, "Response-time analysis in hierarchically-scheduled time-partitioned distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2017–2030, 2016.

[9] E. Hamelin, M. A. Hmid, A. Naji, and Y. Mouafo-Tchinda, "Selection and evaluation of an embedded hypervisor: application to an automotive platform," Proceedings of the 10th Embedded Real-Time Systems International Congress (ERTS 2020).

[10] M. Masmano, I. Ripoll, A. Crespo, and J. Metge, "Xtratum: a hypervisor for safety critical embedded systems," Proceedings of the 11th Real-Time Linux Workshop 2009, pages 263-272.

Adoption of ACPS in Nuclear Reactor Analysis

Christian Castagna
The Unit of Nuclear Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel; email: castagna@post.bgu.ac.il

Daniela Cancila
Université Paris Saclay, CEA LIST, Gif-sur-Yvettes, France; email: daniela.cancila@cea.fr

Antonio Cammi
Politecnico di Milano, Department of Energy, CeSNEF (Enrico Fermi Center for Nuclear Studies),via La Masa 34, 20156 Milano, Italy; email: antonio.cammi@polimi.it

Abstract

Nuclear power plants are a paramount example of critical cyber-physical systems. Some of the current researches in nuclear reactor analysis concern the degree of acceptable uncertainty of the whole system. Some difficulties arise from weaving fields of different disciplines, such as computer science (e.g. embedded software and hardware), mechatronics (e.g. sensors and actuators) and physics (e.g. neutronics and thermal-hydraulics). To complicate the scenario further, each field demands different disciplines, competencies and different teams. In this short paper, we highlight the importance of the cross-fertilization of different disciplines in the nuclear reactor domain and we investigate emerging methods to control uncertainty in the nuclear reactor design.

Keywords: Nuclear Reactors, Safety, Uncertainty, (Autonomous) Cyber-Physical Systems, Machine Learning.

1 Introduction

Since the second half of the last century, we have witnessed deployment of nuclear power plants in the world. They differ with respect to power, design, adopted technologies and digital systems (list not exhaustive). For example, in the seventies, 900 MegaWatt nuclear power plants had a hardwired *Instrumentation and Control* (I&C), while, in 1985, the French 1300 MegaWatt nuclear power plants deployed the first digital I&C system (ref. to DIPS - the Digital Integrated Protection System).

A nuclear power plant is an example of Cyber-Physical Systems (CPS) [1], i.e. physical systems monitored and controlled by electronic systems, otherwise termed I&C systems. In a nuclear power plant, a nuclear reactor is, by namesake, the core of a power plant where nuclear fissions happen. A nuclear reactor is controlled by numerical I&C commands, sent either automatically by the software or directly by the operators in the main control room.

In the I&C systems, the protection system is one of the most critical systems. Its main mission is to control the nuclear reactor in the case of nominal as well as degraded behavior. Its functionality plays a leading role in setting "safety measures to prevent incidents and to mitigate their consequences if they were to occur" [2] - where "incidents include initiating events, accident precursors, near misses, accidents and unauthorized acts (including malicious acts and non-malicious acts)" [2].

To prevent that an accident happens, the nuclear safety authorities demand to assess a safe threshold (or safe margin) and, consequently, a second and depending threshold devoted to the margin of errors (list non-exhaustive). The latter ensures an additional protection measure, which ought to consider acceptable uncertainty. In both cases, the threshold depends on several parameters that involve different countermeasures, i.e. different I&C commands.

Nuclear safety standards IEC 60880 [3] and IEC 61513 [4] define the safety objectives for a safe system and critical software. They demand (A) measures for functional and performance requirements in response to specific signals or conditions. Among the parameters under examination, response time and accuracy have to be taken into account [4]. Moreover, (B), all measures must be validated and proved (list non-exhaustive).

To achieve a greater level of accuracy, we need to control the acceptable uncertainty degree of each component and system integration involved in a nuclear reactor design. All these measures and controls are resources demanding. It constitues an issue, since they must be proved and validatedin a reliable execution time (list non-exhaustive). However, a closer examination reveals that an accurate computational model of a nuclear reactor behavior is too complex to be controlled and simulated in a predefined computing execution time.

In this short work-in-progress paper, we address accuracy for nuclear reactors behavior modeling and we suggest the use of autonomous machine learning algorithms in multi-physics modelling. We conclude with a section on safety nuclear regulations and on-going works.

Before introducing the state of the art and our analysis, let us briefly argue why this research problem is still an important research topic, which demands innovation. As introduced

before, the nuclear safety regulations require accuracy, the identification of the acceptable uncertainty in a reliable execution time. An accurate model of a nuclear reactor behavior is too complex because data arise from different disciplines (e.g. neutronics, thermal-hydraulics, computer science). To complicate the scenario further, data influence each other. In the meantime, the increased hardware performance capability and advanced numerical methods are allowing innovation in multi-physics models to calculate and prove more accurate measures to assess safe margins [5].

2 State of The Art

In recent years, multi-physics simulations of nuclear power plants have become one of the main research topics in the nuclear reactor scientific community [5]. These complex systems are characterized by the interrelation over different physics phenomena, involved in their operation and safety.

Among them, the term *multiphysics* is usually referred to the coupling between neutronics and thermal-hydraulics. The first one is the study of the motion of the neutrons and how they interact with the atomic nuclei [6]. In particular, the fission is the nuclear reaction where a neutron is absorbed by a heavy nucleus, allowing it to split into smaller ones and release a great amount of thermal energy. Indeed, the operation of nuclear reactors is based on the initiation and control of the fission reaction. In neutronics, two main methods exist: deterministic and stochastic ones, this latter based on the Monte Carlo method [6]. Both deterministic and stochastic methods are demanding in execution time, because they have to analyze a very large population of neutrons in space and time as well as the evolution in concentrations of nuclides over fuel. The second one, thermal-hydraulics [6], is the determination of the velocity, density and temperature distribution of the coolant and the fuel[1].

In reactor analysis, neutronics and thermal-hydraulics are two different disciplines and traditionally modelled with single physics computational codes. They solve with different numerical methods the systems of equations, that describe the physical phenomena. However, neutronics and thermal-hydraulics are not independent and they are influenced reciprocally [6]. For example, an increase of the thermal motion of the atomic nuclei, due to a rise of the fuel temperature, allows higher absorption of neutrons (nuclear Doppler effect [6]). Furthermore, the neutronics/thermal-hydraulics mutual interaction determines the power distribution, that is of paramount importance into study the reactor operational and accidental conditions. It follows that the coupling of the physics codes allows improving the accuracy in reactor analysis, necessary with respect to the increase of the safety and thermal efficiency requirements [8].

The current multi-physics couplings usually employ low fidelity models, simulated by deterministic (neutronics) and subchannel codes (thermal-hydraulics), that adopt some approximations of the governing equations for each physics field [5]. An example is the SAPHYR package [9], developed by CEA (*Commissariat à l´énergie atomique et aux énergies alternatives*) [10], characterized by a modular structure composed by different physics codes.

High fidelity and Reduced Order Modelling Recently, high-fidelity models have been developed with Monte Carlo codes for neutronics [11] and Computational Fluid Dynamics (CFD) codes for thermal-hydraulics [12]. These numerical methods are characterized by higher accuracy into reproduce the physics parameters of the reactor, with the possibility to determine locally the quantities of interest. They are also flexible in the implementation with different reactor geometries. In particular, CFD allows characterizing the local condition of the fluid, to study specific thermal-hydraulics phenomena of key interest in safety analysis, as fluid instabilities, turbulent mixing or natural circulation. The simulation of high fidelity models demands a high computational burden, because they use detailed geometry and physics modelling. Therefore, their execution requires the adoption of HPC machines. In the last years, a solution proposed to this problem is the adoption of Reduced Order Models (ROMs) [13]. The latter is referred to a class of advanced numerical methods, where the physical system is well described by a small number of dominant modes, that decrease the degrees of freedom of the original problem.

3 Adoption of ACPS in reactor analysis

Two of the authors have developed in [14] a high-fidelity multi-physics coupling between the Serpent Monte Carlo code [15] for neutronics and the OpenFOAM CFD toolkit for thermal-hydraulics [16]. The modelling approach is validated for the model of the TRIGA Mark II reactor of the University of Pavia, carrying out benchmark analysis with experimental data [13, 17]. In our analysis, in addition to the computational burden, we have observed limits in the process of development and validation of these tools, i.e. the necessity to have detailed experimental data and the *manual* inputs insertion/analysis of uncertainties. The first limit can be overcome in modern nuclear power plants, embedded by advanced electronic I&C systems for monitoring and controlling the reactor [18]. They provide automatic controls for the diagnostic, as well as some processes for operation and safety. Based on these analyses, the reactor operators take some critical decisions.

With respect to the second limit and the problem of the computational cost, we are evaluating to employ autonomous machine learning algorithms with ROM. In the strategy proposed, the original high fidelity multi-physics model is approximated by a ROM (still multi-physics), to be run as many times as required. After that, the underlying idea is to decrease the uncertainties in the input and output parameters through the application of autonomous algorithms, to reach the accuracy needed by the problem under investigation. The proposed methodology is based on the following steps:

1. A (human) designer develops a ROM, obtained by an initial high fidelity multi-physics model of the power plant of interest
2. The model is run by an Autonomous-based Algorithm (AA) to estimate the neutronics (neutron flux) and the

[1] In operating power plants, the fuel has not velocity since it is solid. It is different in the case of the Molten Salt Reactor (MSR) [7], where their innovative design is under study.

thermal-hydraulics parameters (temperature, density, velocity fields)

3. AA compares the simulation outputs with the experimental data obtained by the Data Acquisition System (DAS), computing the error of the model e_m
4. If e_m does not meet the required accuracy, AA adjusts the original ROM, by analyzing the reasons for the discrepancies concerning the neutronics and the thermal-hydraulics parameters of the DAS.
5. The new multi-physics ROM is run by AA and the procedure restarts from step 2.

We point out that the procedure is not only automatic but autonomous, since the role of AA is to understand the deficiencies of the model and improve it via Machine Learning algorithms. With respect to an only human-based driven intervention, AA has the powerful advantage to analyze a huge quantity of information. The expectation is to achieve a better model accuracy and finer control of the acceptable uncertainty degree, required by the nuclear safety regulations.

The use of AA, based on Artificial Intelligence (AI), for the nuclear power plants are under study by the nuclear scientific community and, to our knowledge, at present only in a conceptual fashion (see e.g. the recent works of [19, 20]). In this direction, one of the non-negligible issues to be addressed is to meet the safety requirements, as those provided by the nuclear safety regulations.

4 Nuclear Safety Regulations

In 1979, the TMI-2 accident (Pennsylvania – USA 1979) not only has affected the public opinion but also contributed to rethinking the related nuclear safety standard, the operators' training, and the Nuclear and Regulatory Commissions[2] [21]. Today, we have full separation between the industries, in charge of building and managing a nuclear power plant, and the commissions, in charge of the certification phases and to the inspections during the life of a nuclear power plant.

In Section 1, we referred to two safety regulations for the nuclear domain: IEC 60880 [3], which specifies the safety objectives and requirements for the critical software, and IEC 61513 [4], which concerns safety objectives and requirements for I&C systems and equipment to perform safety functions in a Nuclear Power Plant. These nuclear safety regulations require that all measures, included accuracy, safe margins, uncertainty, response time and execution time (list non-exhaustive), must be validated and proved [4]. This point represents a potential conflict with the adoption of AI for nuclear reactor models, as we have suggested in Section 3. One of the main blocking points is the (formal) proof. Although, nuclear safety regulations are among the more conservative safety regulations and, hence, they not allow AI today, some overcoming solutions could investigate the use of AI within a predefined and controlled context, especially for application

[2] In 1986, the Chernobyl accident (Ukraine) happened during the Cold War. A set of human errors, included the design of the power plant, are the main causes of the accident. The Chernobyl nuclear power plant is not compliant with the safety nuclear regulation.

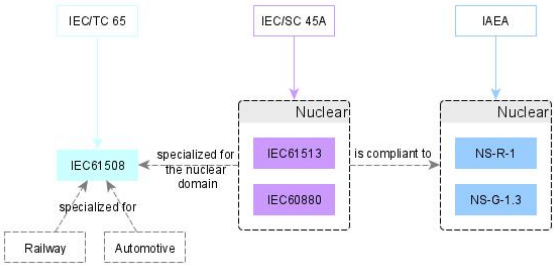


Figure 1: Relations between the safety regulations and the Technical Committees. Legend in the table below.

IEC60880	Nuclear Power Plants - Instrumentation and Control Systems Important to Safety - Software aspect for Computer-Based Systems Performing Category A Fuynctions
IEC61513	Nuclear power plants – Instrumentation and control important to safety – General requirements for systems
IAEA	International Atomic Energy Agency
NS-R-1	Safety of Nuclear Power Plants: Design
NS-G-1.3	Instrumentation and Control Systems Important to Safety in Nuclear Power Plants
IEC61508	Functional safety of electrical/electronic/programmable electronic safety-related systems

in the design phases and for the study of the parameters based on the fuel consumption.

In our study, we plan to perform a test phase, with the Verification and Validation (V&V) of the proposed methodology in the benchmark problems, used for V&V of nuclear reactor codes. In this analysis, we also include the validation of the available experimental data of the TRIGA Mark II reactor. After that, we aim to extend the procedure in codes for industrial application. In addition, a comparison between the results and those come from the traditional techniques (which define a baseline) is mandatory.

For the sake of completeness, Figure 1 highlights some relations between the cited regulations and other regulations (list non-exhaustive). First, it shows that they have to be compliant to the regulations provided by the International Atomic Energy Agency (IAEA). Second, they specialize IEC 61508 [22] for the nuclear application sector. IEC 61508 plays the role of umbrella for different applications domains, such as nuclear, railway and automotive. Although we should be wary of any generalization, IEC 61508 allows us to compare and translate -if it is the case- different solutions and methods to control the degree of acceptable uncertainty of a system among different application domains.

Finally, Figure 1 highlights the Technical Subcommittees in charge of specifying safety regulations. It involves that several experts, from industries and research centers in the world, work together via independent teams to establish a set of requirements to achieve a safer system. Such an organization has the main advantage to be large enough to have a consensus and ensure the independence of the work with respect to a single national interest.

5 Conclusions and Outlook

In this short paper, we briefly discussed the problem to control uncertainty in nuclear reactors. Intriguing solutions are provided by coupling neutronics and thermal-hydraulic

disciplines, otherwise referred as multiphysics approaches. To solve the equations by meeting the nuclear safety constraints [3, 4], simulations of high fidelity models require HPC hardware [6] and consequently the detection and control of all possible errors arising from the HPC itself. We highlight that some human decisions are taken over these results. Then, we arise the necessity of advanced study in computer science to be applied for nuclear reactor analysis. Results promoted by the PROXIMA project on the probabilistic real-time execution [23] on multi- and many-core, for example, could be implemented to the nuclear domain - of course with additional proofs and all possible cautions.

In this paper, we suggest the way forward in an investigation on autonomous algorithms, based on machine learning, to estimate the neutronics (neutron flux) and the thermal-hydraulics parameters (temperature, density, velocity fields) in high fidelity multi-physics modelling. The expectation is to avoid or, *à défaut* reduce, human errors in the decision-making by reducing the uncertainties in the input and output parameters, during the improvement and validation of the multi-physics analysis. This approach involves autonomous and not only automatic I&C. This is why we are very cautious and we stress the importance of the cross-fertilization domain between the computer science and nuclear reactor physics community, even more than in the past. Our on-going work is then to prove each step, compare autonomous-based solutions with more traditional approaches, including accuracy of the model under study, and a comparison of functional and performance requirements in response to specific signals or conditions, as required by the nuclear safety regulations.

References

[1] M. D. Franusich, "Security Hardened Cyber Components for Nuclear Power Plants," Tech. Rep. Grant No. DE-SC0013808, US Department of Energy, Office of Science, Chicago Office, 2016.

[2] International Atomic Energy Agency, *IAEA Safety Glossary*. 2018.

[3] IEC, *IEC 60880 Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category A functions.*, 2006.

[4] IEC, *IEC 61513. Nuclear power plants – Instrumentation and control important to safety – General requirements for systems*, 2011.

[5] J. Wang, Q. Wang, and M. Ding, "Review on neutronic/thermal-hydraulic coupling simulation methods for nuclear reactor analysis," *Annals of Nuclear Energy*, vol. 137, p. 107165, 2020.

[6] R. E. Masterson, *Introduction to Nuclear Reactor Physics*. CRC Press, 2018.

[7] G. 2016, "Generation IV International Forum (GIF) Annual Report 2014," tech. rep., 2015.

[8] J. Portugal-Pereira and et al., "Better late than never, but never late is better: Risk assessment of nuclear power construction projects," *Energy Policy*, vol. 120, no. May, pp. 158–166, 2018.

[9] B. Akherraz and et al., "SAPHYR: A Code System From Reactor Design to Reference Calculations," in *Proceedings of M&C 2003*, (Gatlingburg, USA), p. 7, April 6-11, 2003.

[10] http://www.cea.fr/.

[11] I. Lux and L. Koblinger, *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*. Boca Raton, Florida: CRC Press, inc., 1991.

[12] S. B. Pope, "Turblent Flows," 2001.

[13] C. Castagna, *A Multi-Physics Modelling Approach for the Analysis of Burnup Calculation of the Next Generation of Nuclear Reactors*. PhD thesis, Politecnico di Milano, 2020.

[14] C. Castagna, E. Cervi, S. Lorenzi, A. Cammi, and al., "A Serpent/OpenFOAM coupling for 3D burnup analysis," *European Physical Journal Plus*, vol. 135, no. 6, 2020.

[15] J. Leppänen and et al., "The Serpent Monte Carlo code: Status, development and applications in 2013," *Annals of Nuclear Energy*, vol. 82, pp. 142–150, 2015.

[16] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A tensorial approach to computational continuum mechanics using object-oriented techniques," *Computers in Physics*, vol. 12, no. 6, p. 620, 1998.

[17] C. Castagna, M. Aufiero, S. Lorenzi, G. Lomonaco, and A. Cammi, "Development of a reduced order model for fuel burnup analysis," *Energies*, vol. 13, no. 4, pp. 1–26, 2020.

[18] IAEA, "Advanced Surveillance , Diagnostic and Prognostic Techniques in Monitoring Structures , Systems and Components in Nuclear Power Plants," Tech. Rep. NP-T-3.14, 2013.

[19] D. Lee and J. Kim, "Autonomous Algorithm for Safety Systems of the Nuclear Power Plant by Using the Deep Learning," in *Proceedings of the AHFE 2017*, (Los Angeles, California, USA), 2017.

[20] J. Kim, D. Lee, J. Yang, and S. Lee, "Conceptual design of autonomous emergency operation system for nuclear power plants and its prototype," *Nuclear Engineering and Technology*, vol. 52, no. 2, pp. 308–322, 2020.

[21] J. Samuel Walker, *Three Mile Island: A Nuclear Crisis in Historical Prospective*. 2004.

[22] IEC, *61508:1998 and 2000, part 1 to 7. Functional Safety of Electrical, Electronic and Programmable Electronic Systems.*, 2000.

[23] PROXIMA, "Probabilistic real-time control of mixed-criticality multicore and manycore systems, ecsel eu project."

Auto-generated Coherent Data Store for Concurrent Modular Embedded Systems

James S. Kimmet

Raytheon Missiles & Defense, 1151 E Hermans Rd, Tucson AZ 85706, USA; email: jskimmet@raytheon.com

Abstract

A thread-safe data store has been developed to enforce interface consistency and shared data coherency in a concurrent modular embedded real-time system. Typical messaging techniques may not provide optimal data transfer between software components in all embedded systems, especially if there is a high degree of data interdependency as the number of components increases. The data store paradigm reduces the overall communication load by providing finer data item granularity, and eliminating the copy and transfer of unused message content. The data store described in this paper is implemented with code auto-generation and provides compile-time error checking, ensuring effortless data integrity by automatically rebuilding when software component interfaces are changed. The data store has been successfully employed to rehost a highly-coupled legacy software application into a more modularized component architecture.

Keywords: modular embedded systems, code auto-generation, concurrency, multicore, pipelined data transfer.

1 Introduction

Many embedded software systems rely on messaging techniques for data transfer between software components. These push-based communications work well in event-driven real-time systems, and ideally can provide a reasonable degree of decoupling. For example, in a publish/subscribe system, the data producer can remain entirely agnostic to the number and specificity of its consumers [1]. The mechanics of output data delivery to the consumers is handled separately by a message routing subsystem, isolating the producer and consumer components from knowledge about or internal changes to each other's implementation details.

One potential drawback to message-based data transfers, however, is the large-scale granularity of the transfer structures. In one extreme, each software component produces a single output message at the end of its process, containing the various data items it has calculated that may be needed by other components in the system. Each component that needs one or more of these data items receives that component's single output message as input for its process. It is highly unlikely, however, that all consumer components of a given output message actually use every data item in that message. The single output message necessarily contains a superset of the data items needed across all the consumers in the system, so each consumer typically receives superfluous content in the message that it doesn't actually utilize. In this case, the message routing system copies and transfers unused data to each consumer, representing an inefficient use of processor and memory resources.

In the other extreme, each software component generates numerous output messages, each containing only a single data item. This approach ensures that each consumer component receives only the specific inputs it needs for its process. There is no wasteful copying or transfer of unused data, but the sheer volume of message distribution can overwhelm the routing subsystem as the number of components scales up in larger systems. A balance is therefore required. One mechanism derives some small number of output messages per component, each containing some meaningful aggregate of related data items that have common consumers. Optimizing this approach requires overt knowledge of the various consumers of each component's output, which violates the objective of creating a decoupled, extensible, agnostic data transfer facility.

The balancing approach between wasteful copying of unused message content and excessive routing of numerous diminutive messages can become unmanageable as the component scope and number increase. Consider rehosting of a legacy tightly-coupled system in which each component produces a large amount of data, different subsets of which are required by a large number of other components in the system. Such a system has low cohesion [2], and a push-based data transfer model may not even be feasible without overloading the messaging subsystem [3].

Figure 1 shows a simplified example of such a system in which just four components each produce output data required by several other components. The output data are shown as a colored box within each component, which represents internal state that remains persistent until the next iterative execution of that component. In this example, input data are actively retrieved by each consumer component, by calling multiple public accessor functions provided by each producer (*i.e.*, pull-based communication), rather than by messaging.

The legacy system this example represents is actually much more complex, consisting of dozens of components, each maintaining scores of internal state data items. Extensive analysis of the many-to-many data relationships in this system revealed that refactoring these active data retrievals

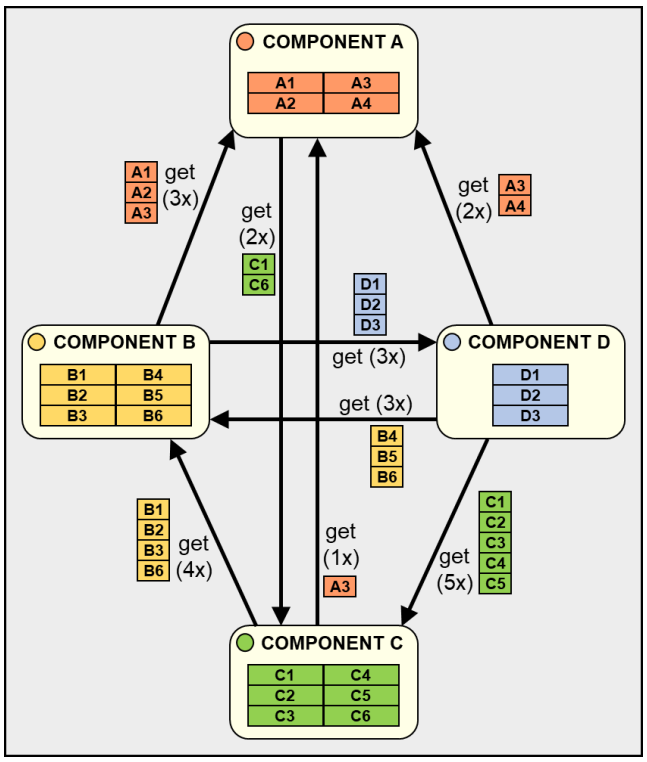


Figure 1: Pull-based data fetches in a tightly-coupled system with low cohesion

into a push-based messaging technique was untenable. Rather than forcing a complete restructure of the legacy system's architecture, simplification of the system is achieved by consolidating the various state data items into a centralized storage module. This approach accomplishes the desire of decoupling the various rehosted software components, and results in a more robust overall system design, providing coherent data transfers through a push-pull data exchange.

The approach presented is a thread-safe data store, which enforces global data coherency in a concurrent modular system. The data store is implemented with code auto-generation, ensuring data integrity and interface consistency that automatically updates when software component interfaces are changed. The auto-generation process also provides built-in error checking and creates a comprehensive data usage report at compile time, identifying the producer and consumers for each data item for offline analysis. The described solution further incorporates an iterative run-time state storage feature, which provides access to multi-stage state history for pipelined processing systems. This data store has been successfully employed to rehost a highly-coupled legacy software application into a more modularized component architecture, and is currently being integrated into multicore real-time embedded hardware.

2 Global data store design

The global data store is a hybrid approach which combines the advantages of a push-based mechanism (allowing each component to be agnostic to the consumers of its output) and the pull-based approach (supporting the finer granularity for each component to retrieve only the data items it needs as input).

Figure 2 shows the previous example rehosted using the global data store to manage all the output data. Note that the colored boxes representing each component's output state have been collected into a single repository, and each component transacts directly with the common data store for both input and output. The components themselves are completely decoupled from each other in this rehosted system, improving the modularity of the implementation.

As each component is activated for processing, it performs a *single* data retrieval operation from the data store, rather than calling *multiple* accessor functions on several components as was done in the legacy system. At the conclusion of processing, the component performs a *single* data transfer operation back to the data store, where the outputs persist until they are again refreshed after the next processing iteration. This push-based output operation isolates the component from knowledge of its consumers, and does not require any attempt to aggregate the data items into multiple subset groupings. The pull-based input operation retrieves just the required input data items, and does not waste resources copying unneeded data items produced by the other components. In other words, the retrieval operation is not forced to copy any producer's entire output message, but rather selects only the items specified as required inputs by the consumer component's interface, as a single fetch operation.

The global data store accessor functions employ a concurrency control (specifically a mutex semaphore with priority inheritance) to support thread-safe data transfer operations between components in a multi-threaded real-time embedded system. The critical operations protected by the mutex consist of simple memory copy instructions, minimizing the timing impact on competing threads. By collecting its various input data items into a single fetch of

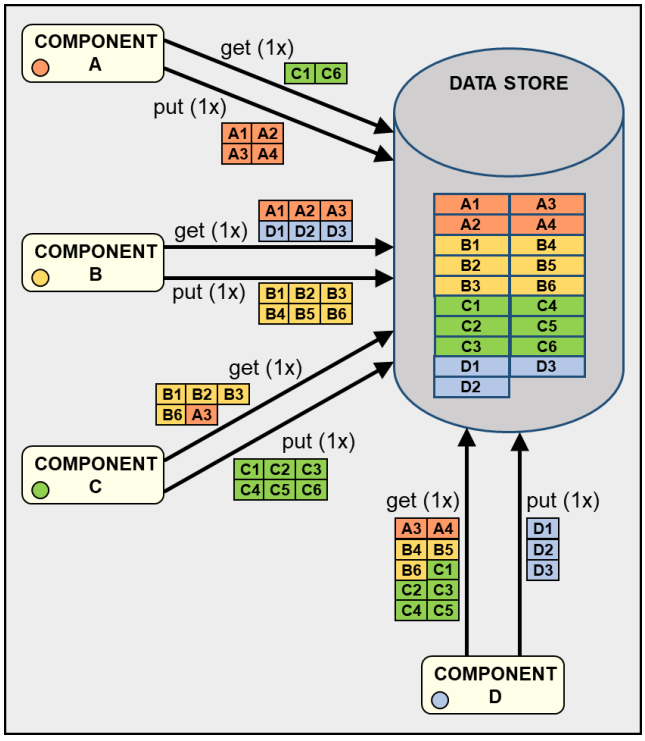


Figure 2: Previous example rehosted with a global data store

an aggregate structure, each component receives a coherent snapshot of the relevant system state at the beginning of its processing iteration. This is a clear advantage over the legacy use of multiple accessor calls, especially in a multi-threaded system, in which the overall system state might change during the course of the various legacy input fetches, resulting in an incoherent set of inputs used for processing.

The data store design also incorporates an additional feature to provide multi-stage state history storage for iterative pipelined processing systems. As shown in Figure 3, the state history feature provides a configurable depth of system run-time state, which can be accessed for read/write operation at any specified level as needed by each component. The state history is implemented as a circular buffer, with a rotating index indicating the current state. This feature enables components to recall information regarding previous iterations of the system state that may be required while processing later iterations, and eliminates the need for multiple components to each implement their own internal persistence of pipelined run-time state.

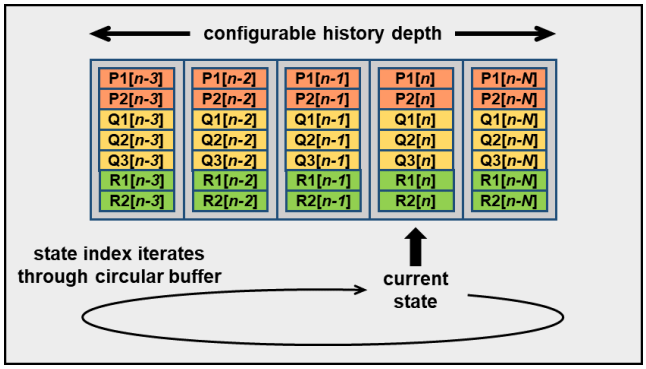


Figure 3: Pipelined state history storage for iterative processing systems

Since the global data store requires the use of shared memory for the components interfacing with it, it is not directly applicable to data transfers between components that lack a shared memory model (*e.g.*, those deployed in multiprocessor systems located on separate circuit cards). In such a system, however, proxy components [4] within the shared-memory region can be employed to act as intermediaries between the data store and the remote components hosted on the other processors.

3 Code auto-generation

One of the primary issues during extended development of a modular software system is the continual maintenance of the data transfer mechanism. The key to ensuring data integrity is to maintain consistent interface definitions across all components as the system evolves. To mitigate this concern, the data store uses code auto-generation to automate the data storage and transaction implementations. The data store code automatically rebuilds whenever there is a change in a component's interface definition, ensuring consistency of the store with the latest interface specification and eliminating the risk of introducing human error during numerous system development iterations.

The code generation process is controlled by the definition of input and output structures for each software component. These are the same C++ structure definitions used by the component itself for handling the interface data items, and are typically created by each component developer. The code generator uses custom *flex* and *bison* tools [5] to parse each of the C++ interface definition files, building a master list of all output data items for the overall software system, as well as the specified input and output aggregations of these items. The generated code consists of the data store class, which includes each of the individual data items as a private member variable, along with public setter and getter functions corresponding to each of the interface-specified input/output structures. The functional implementation of each setter/getter function is simply a group of mutex-protected copy expressions.

The real advantage of the code generation technique is not just in the enforced consistency of the data store implementation, but in the additional automated capabilities it provides. As part of the parsing activity, the code generator must match each input item to an existing output item to ensure data availability. Name or data type mismatches result in code-generation errors and compilation failure. To protect data integrity, each data item in the store is allowed to have only a single producer; thus, multiple output structures found to contain the same data item likewise result in a compilation error. The rule set defined within the flex/bison toolchain can be customized to enforce architectural policies and coding standards (*e.g.*, naming conventions, type restrictions). Upon successful compilation, the code generator produces a comprehensive data usage report, listing each data item by name and data type, along with its producer component and a list of all consumer components. These usage metrics can be used to monitor data interface trends during system development, and for predictive analysis regarding system performance.

4 System integration results

The auto-generated data store design has been successfully applied to refactor the example legacy system discussed above. Component developers conducted an interface analysis of each software component, deriving explicit definitions of interface structures to be used for data store transactions. Input structures were determined by identifying all input data items the component manually retrieved via accessor calls in the legacy implementation. Output structures were specified to contain all internally computed data items that were previously provided by the component's own public accessors.

The resulting set of interface structure definitions was used to feed the initial data store code generation process, producing a working data transfer mechanism available from the onset of the rehosting effort. Since the rehosting project progressed incrementally from a minimal viable product, there were often cases in which data items needed as inputs for a component were not available from a producer. To prevent compilation errors during the code generation, a surrogate data producer interface was defined. The surrogate producer temporarily defined the data item and its

initialization value in the data store, until the actual producer component was integrated into the rehosted application. Through multiple continuous-integration iterations, component developers updated their components' interface definition files, adding new features, clarifying variable names, integrating new composite data types, etc. With each change, the data store code was automatically regenerated during the build process, ensuring that producer and consumer data definitions remained synchronized. Any mismatches prevented compilation and required immediate resolution of the offending change.

The rehosted legacy system is still being expanded. Currently it consists of 72 software components interfacing through the data store mechanism. The auto-generated data store contains 715 data items that are shared and transferred between these components. Each individual data item has just a single producer component to ensure coherency, but in the current implementation is distributed to as many as 24 consumers.

The rehosted software has been integrated in a simulation environment, which can exercise the entire system in numerous deployment scenarios to provide wide coverage of each component's functionality. The simulation models the multiple threads of the embedded software system being triggered at various nonsynchronous rates, although pre-emptive scheduling is not modelled. (Tasks are allowed to run until blocked, prior to context switching.) The simulation has provided an excellent development environment to implement and test the data store design while incrementally making progress rehosting the application logic from the legacy software. Within this simulation application, the data store has demonstrated consistent performance, transferring data items efficiently between components and maintaining a coherent system state.

5 Conclusions and future work

This global data store design has been implemented and integrated into a functioning software system, facilitating the rehost of existing legacy application logic into a more cohesive, modular, extensible software architecture. The code auto-generation capability has proven invaluable in maintaining working component interfaces with minimal human burden and ensuring the continued operation of the rehosted system throughout the incremental development project. The data usage reports automatically generated during the build process provide useful insight into the performance of the data transfer mechanism, and the dependencies across the system upon particular data items.

Currently, the legacy rehost project described above is continuing with integration into embedded hardware containing multicore processor devices. The multicore architecture employs a symmetric multiprocessing (SMP) configuration, allowing the threads allocated to each core to share common memory resources. With its built-in concurrency protections, a single global data store inherently supports data transfers between all components allocated across the multiple cores within each processor device.

The global data store described in this paper is an efficient alternative to push-based message transfers for shared-memory real-time embedded systems. Similar to publish/subscribe messaging subsystems, use of the data store as the data delivery mechanism allows the complete decoupling of each software component from all others. The finer granularity of the data accesses from the data store, however, along with the coherency of fetching all inputs with a single transaction, provide distinct advantages over message-based systems. Concurrency controls ensure data integrity within multi-threaded data store applications. Code auto-generation automatically keeps the data store implementation updated when changes occur in component interfaces, and enhances system robustness with automated error-checking and interface consistency verification, supporting iterative software development projects.

References

[1] S. Tarkoma, *Publish/Subscribe Systems: Design and Principles*, John Wiley & Sons, 2012.

[2] S. Husein, A. Oxley, "A coupling and cohesion metrics suite for object-oriented software," *IEEE International Conference on Computer Technology and Development*, 2009.

[3] Z. Jerzak, C. Fetzer, "Handling overload in publish/subscribe systems," *IEEE International Conference on Distributed Computing Systems Workshops*, 2006.

[4] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[5] J. Levine, *flex & bison: Text Processing Tools*, O'Reilly Media, 2009.

M2OS for Arduino Uno: Ada Tasks and Arduino Libraries Working Together

Mario Aldea Rivas, Héctor Pérez Tijero

Universidad de Cantabria, Avda. Castros s/n, Santander, 39005, Spain; email: {aldeam,perezh}@unican.es

Abstract

This paper presents the current status of the porting of M2OS to the Arduino Uno board. M2OS is a small real-time operating system targeted to microcontrollers with very tight memory constraints. M2OS provides support for a simple Ada tasking model based on non-preemptive one-shot tasks. The paper puts special emphasis on the adaptation of the standard Arduino core library to be used by Ada applications running on M2OS.

Keywords: Ada, Tasking, Arduino Uno, M2OS.

1 Introduction

The tasking constructs provided by the Ada language require a large run-time system to support them. Even considering the reduced subset included in the Ravenscar profile, the memory requirements of the Ada tasking are excessive for the limited memory available in small microcontrollers.

With the aim of bringing a simplified Ada tasking to microcontrollers with very tight memory constraints we have developed M2OS [1][2]. M2OS is a small real-time operating system that supports the non-preemptive one-shot tasking model.

M2OS is distributed as free software [1] and it currently provides support for three targets: STM32f4 board (based on the ARM Cortex-M4 microcontroller), Epiphany many-core and Arduino Uno.

The Arduino project[2] provides open hardware and software platforms with the objective of easing the building of digital devices. One of the most popular boards of the Arduino family is Arduino Uno. This board is based on the 8-bit ATmega328P AVR microcontroller, a very modest computing platform with only 2KB of SRAM memory and 32KB of Flash.

One remarkable reason behind the popularity of the Arduino technology is the availability of a large number of standard[3] and third-party libraries. Those libraries include low-level drivers to access a large number of devices (sensors and actuators). The libraries also include higher-level modules such as some communication protocol stacks.

Typical Arduino Uno applications are built as sequential C/C++ programs. There are some solutions that provide limited concurrency capabilities on the Arduino Uno board, one of the most complete being ARTe (Arduino Real-Time extension) [3].

From the Ada perspective, a very interesting project applicable to the Arduino Uno board is AVR-Ada [4]. AVR-Ada provides developers with a compiler, a run-time system (RTS) without tasking support (i.e., the zero footprint run-time) and a set of device support libraries written in Ada.

Despite this remarkable effort, it is really hard for the Ada community to develop libraries for the huge amount of Arduino devices that are constantly launched to the market. To overcome this issue, this work opts for making the existing C/C++ libraries available to Ada applications running on top of M2OS. Since most of the third-party libraries rely on the Arduino core library, the main objective of this work is the integration of this core library with M2OS.

2 M2OS for Arduino Uno

2.1 M2OS

M2OS implements a scheduling policy based on the non-preemptive one-shot task model. One-shot tasks do not keep any local status between activations. That behaviour allows sharing the same stack area among all the tasks and therefore, the system only needs to allocate a stack area large enough to fit the largest task stack. This feature is crucial in microcontrollers with very tight memory constraints such as the Arduino Uno.

One-shot tasks are created using a predefined task type (One_Shot_Task) defined in the M2OS specific package AdaX_Dispatching_Stack_Sharing. An example of one-shot task is shown in Listing 1.

```
User task data;

procedure User_Task_Init is
begin
  Initialization Instructions;
end User_Task_Init;

procedure User_Task_Body is
begin
    Task Job Body Instructions;
    Next_Time := Next_Time + Period;
    delay until Next_Time;
end User_Task_Body;
```

[1] https://m2os.unican.es/
[2] https://www.arduino.cc/
[3] https://www.arduino.cc/en/reference/libraries

```
User_Task :
  AdaX_Dispatching_Stack_Sharing.One_Shot_Task
    (Init_Ac  => User_Task_Init'Access,
     Body_Ac  => User_Task_Body'Access,
     Priority => 10);
```

Listing 1: One-shot task creation using the M2OS API

Under this approach, the first activation of the one-shot task will execute the User_Task_Init procedure, and subsequent activations will trigger the execution of User_Task_Body. To verify the one-shot model, the last instruction of the User_Task_Body procedure must be a dispatching operation, i.e., a delay until (as in Listing 1), an entry of a protected object or a Suspend_Until_True operation on a suspension object.

Alternatively, one-shot tasks can also be created as standard Ada tasks as long as they fulfil the structure shown in Listing 2.

```
task User_Task with Priority => 10;
task body User_Task is
  User task data;
begin
  Initialization Instructions;
  loop
    Task Job Body Instructions;
    Next_Time := Next_Time + Period;
    delay until Next_Time;
  end loop;
end User_Task;
```

Listing 2: One-shot Ada task structure

It is important to note that the source code shown in Listing 1 and Listing 2 are functionally equivalent. For the latter, M2OS includes a code transformation tool based on libadalang [5] that transforms Ada tasks according to the structure shown in Listing 2 to M2OS code similar to Listing 1. Further details on M2OS and its scheduling policy can be found in [1].

2.2 M2OS port to Arduino Uno

The layer view of M2OS running on the Arduino Uno board is shown in Figure 1. Ada applications rely on a simplified RTS which is based on the Ravenscar RTS provided by AdaCore. The interaction between the RTS and M2OS is performed through the operating system API. Furthermore, M2OS has a Hardware Abstract Layer (HAL) that comprises the interaction of the kernel with the underlying hardware platform. The HAL includes operations for interrupt management, time accounting, context switch, stack usage accounting and basic console output.

The implementation of the HAL layer for the ATmega328P is mostly straightforward due to the simplicity of this microcontroller. The most complex part is the implementation of the M2OS system timer, which is briefly detailed next.

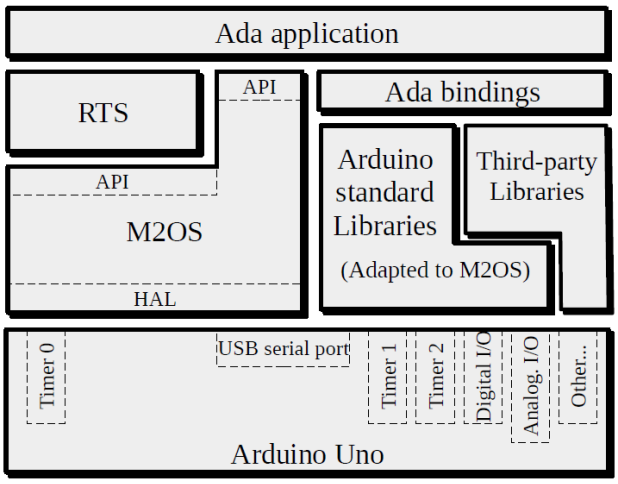


Figure 1: Layer view of M2OS on Arduino Uno

The ATmega328P microcontroller counts on 3 hardware timers (see Figure 1). M2OS takes over the Timer 0 and programs it to generate a periodic interrupt each 1 ms. The associated interrupt handler is in charge of maintaining the system time and executing the timing events when their time is reached. Since Timer 0 is used by M2OS, it is necessary to rewrite the time-related functions of the Arduino core library. Further details about this modification will be provided in Section 3.2.

Besides, the M2OS HAL includes a simple driver for the Arduino USB serial port. Despite a driver for the USB serial port is already included in the Arduino library, it has also been implemented in M2OS to keep it independent from the Arduino library.

The development process of applications for M2OS is based on Linux and the gnatstudio IDE. We have built from sources a cross avr-gcc compiler with support for C, C++ and Ada languages. To make the development process user-friendly, gnatstudio has been customized to include buttons to build and download the application to the Arduino Uno board (using the AVRDUDE[4] command). Furthermore, the Arduino serial console has been integrated in the IDE as a console window, and an Arduino Uno emulator (SimulAVR[5]) has also been integrated with gnatstudio.

3 Arduino libraries in M2OS

3.1 Arduino libraries

In the Arduino programming model (based on C and C++ languages), the API of the standard Arduino libraries is available to applications through a set of C and C++ header files. The basic functionality, implemented by the core library, is provided in the Arduino.h file, and it includes digital and analog I/O, time-related functions, and some utility functions.

The remaining standard Arduino libraries, which are devoted to more specialized purposes, are available to applications through other header files like Servo.h (for servo motor

[4] https://www.nongnu.org/avrdude/
[5] https://www.nongnu.org/simulavr/

control), Wire.h (for communication with I2C/TWI devices), Ethernet.h, WiFi.h, etc.

It is important to remark that the Arduino Uno implementation of the standard libraries uses the three timers provided by the ATmega328P: while Timer 0 is used to implement the time-related functions, timers 1 and 2 are used for servo pulse-width modulation and sound wave generation, respectively.

Apart from the standard libraries, every device in the Arduino ecosystem brings its own libraries developed by the device makers and/or by third-party programmers. These device libraries usually rely on the functionality provided by the standard libraries. The Arduino IDE provides a mechanism to install non-standard libraries. Once libraries have been installed, their source files reside in subfolders inside the IDE "Library folder".

To make standard and third-party Arduino libraries available to Ada applications, the core library must be adapted to run on M2OS and, therefore, the rest of standard and third-party libraries built on top of it must be recompiled. Additionally, a set of Ada bindings are also required as shown in Figure 1. Each step of this process is briefly described in the following sections.

3.2 Modifications to the Arduino standard library

Most of the Arduino libraries consist of simple low-level code that can be used by Ada applications as long as the appropriate bindings are available.

As stated in Section 2.2, the only modifications required to integrate the Arduino libraries with M2OS are related to the use of the Timer 0 and these modifications only affect to the core library. The Timer 0, which is needed by M2OS to implement its system clock, is used by the Arduino library to support the time-related functions delay(), millis() and micros() (these functions are defined in the file wiring.c).

As a result, the delay(), millis() and micros() functions have been rewritten for M2OS. Special care has been taken on keeping the same functionality than the original Arduino functions since most libraries rely on them.

The Arduino function delay(ms) pauses the program execution for the specified number of milliseconds. In our adaptation, this function performs an active waiting on the M2OS system clock until the desired number of milliseconds has elapsed. Notice that the use of an active waiting is intentional in order to avoid that code from the Arduino libraries calling delay() could cause the suspension of the running Ada task (one-shot tasks ends its current activation once a dispatching point is reached). Instead of using delay(), Ada tasks should use the delay until primitive to suspend themselves.

Function millis() returns the number of milliseconds since system startup. In the M2OS adaptation, this function simply returns the number of milliseconds measured by the M2OS system clock.

Function micros() returns the number of microseconds since system startup. In our modification, this function returns the milliseconds measured by the M2OS system clock (multiplied by 1000), plus the current count of Timer 0. This implementation provides a resolution of 4 microseconds (the duration of the Timer 0 tick).

Notice that the fourth Arduino time-related function, delayMicroseconds(), does not need to be modified since its standard implementation is based on an active waiting using loops.

Another aspect to emphasize is that there is no need to include any synchronization primitive in the Arduino libraries to make them task-safe, as the non-preemptive policy used by M2OS avoids concurrent accesses to them.

3.3 Building the Arduino library for M2OS

The standard process of building an Arduino program[6] (a .ino file called *sketch* in Arduino terminology) is carried out by the Arduino IDE. Along with the *sketch* file to be built, the Arduino IDE compiles all the .c and .cpp files in the *sketch* folder. The IDE also builds an instance of the Arduino library that includes all the dependences of the *sketch* and the other .c and .cpp files.

Dependencies are solved by recursively looking for the header files included by the *sketch* and the other .c and .cpp files. For each included header file, the .c and .cpp files in the corresponding library folder are also compiled and added to the library.

Taking advantage of the aforementioned Arduino building process, M2OS provides an automatized mechanism based on a Makefile to build the adapted M2OS Arduino library.

For each library supported by M2OS, the Makefile checks if it is installed in the "Library folder". A header file which includes all the supported libraries that are installed in the system is automatically generated.

Besides, the M2OS modified version of the wiring.c file (containing the adapted functions delay(), millis() and micros()) is included in the *sketch* folder.

An empty *sketch* that includes the automatically generated header file is then built using the Arduino IDE through the command line. The outcome of this building command is a folder that contains the object files of the files in the *sketch* folder and the libraries included by them.

Then, the Makefile creates a library file containing the generated objects files. Additionally, the Makefile ensures that the compiled version of the wiring.c file included in the library is the M2OS adaptation instead of the original one.

Finally, the resulting library is added to the linking command of the M2OS applications.

3.4 Ada binding to the Arduino libraries

The Ada bindings shown in Figure 1 are organized forming a package hierarchy. The root of that hierarchy is the

[6] https://arduino.github.io/arduino-cli/latest/sketch-build-process/

package Arduino, which includes the functionality defined in the standard header file Arduino.h. Bindings for other standard headers are provided as child packages, i.e., Arduino.Servo (binding for Servo.h), Arduino.Wire (binding for Wire.h), etc.

We have chosen a "thin binding" strategy where the Ada packages make direct calls to the C/C++ functions without managing the internals of the C/C++ types or classes. In particular, C++ classes have been defined as opaque Ada records having as only field an array of bytes of the required length to match the C++ class size (reported by the sizeOf operator).

Apart from the standard header files, M2OS also includes bindings for some popular Arduino devices such as infrared thermometer, I2C LCD display, temperature and humidity sensor, ultrasonic ranger, voice recognizer, etc.

We have also developed more advanced modules that takes advantage of M2OS tasking to perform activities very hard to implement in a sequential program. For example the Arduino.Buzzer package includes a priority configurable task to play melodies with a piezoelectric buzzer, and the Arduino.PIR package includes a task with configurable period and priority to count the number of motion detections performed by a PIR Sensor.

4 Evaluation metrics

The memory footprint is an essential metric when working with microcontrollers with tight memory constraints such as the Arduino Uno. Table 1 shows the sizes of the applications as reported by the avr-size command.

	text	data	bss	total
Blink (Original Arduino Sketch)	*924*	*0*	*9*	*933*
Blink (periodic task)	3408	202	40	3650
Two periodic tasks	3176	192	78	3446
Six periodic tasks	3744	192	230	4166

Table 1: Size of applications (bytes)

The first row in Table 1 shows the size of the "blink" Arduino IDE example, a very simple sketch that turns on and off a led integrated in the board. The second row shows the size of a functionally equivalent M2OS application that uses an Ada task to blink the led. These values show that M2OS adds 2686B in Flash (the difference in the text and data sections) and only 31B in RAM (the difference in the bss section).

The last two rows in Table 1 show the size of two stub applications with two and six periodic tasks, respectively. By comparing them, we can deduce that the memory required by each task stub is 142B in Flash and 28B in RAM.

We consider that the obtained memory footprint values are particularly acceptable, especially when considering the benefits that the multitasking capabilities provided by M2OS can offer to Arduino Uno programmers.

5 Conclusions and future work

M2OS allows developing simple multitasking Ada applications (based on the non-preemptive, one-shot model) on the Arduino Uno board.

A user friendly developing environment, based on gnatstudio, automatizes the building and downloading of applications. The environment also integrates the system console and an Arduino Uno emulator.

The Arduino core library has been modified to be compatible with M2OS. Since most third-party libraries rely on the core library, its adaptation allows M2OS applications to use the huge number of libraries existing in the Arduino environment.

M2OS currently provides a set of Ada bindings for many standard and third-party libraries. However, some interesting functionality is not supported yet. For example, support for the Ethernet.h and WiFi.h standard header files in M2OS would be of interest for a wide range of projects.

Another open line of work is including support for more devices and popular Arduino Uno robot kits.

We also plan to develop complex applications to validate M2OS and its integration with the Arduino libraries. Beside, these complex applications will allow to show the advantages that concurrent programming could bring to the Arduino Uno world.

References

[1] M. Aldea Rivas and H. Pérez Tijero, "Leveraging real-time and multitasking Ada capabilities to small microcontrollers", *Journal of Systems Architecture*, Volume 94, Pages 32-41, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2019.02.015, 2019.

[2] M. Aldea Rivas and H. Pérez Tijero, "Proposal for a new Ada profile for small microcontrollers", *Ada Letters*, 38(1):34–39, doi:10.1145/3241950.3241955, 2018.

[3] P. Buonocunto, A. Biondi, M. Pagani, M. Marinoni, G. Buttazzo, "Arte: Arduino real-time extension for programming multitasking applications", *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, ACM, 2016, pp. 1724–1731, 2016.

[4] R. Ebert, AVR-ADA: Ada cross compiler and libraries for AVR µCs, URL https://sourceforge.net/p/avr-ada/

[5] R. Amiard and P. De Rodat, "Easy Ada Tooling with Libadalang", Free and Open source Software Developers' European Meeting (FOSDEM).

Fuzion – Safety through Simplicity

Dr. Fridtjof Siebert
Tokiwa Software GmbH, Karlsruhe, Germany; email: siebert@tokiwa.software

Abstract

Fuzion is a modern, general purpose programming language that unifies concepts found in structured, functional and object-oriented programming languages into the concept of a Fuzion feature. It combines a powerful syntax and safety features based on the design-by-contract principle with a simple intermediate representation that enables powerful optimizing compilers and static analysis tools to verify correctness aspects.

Fuzion maps different concepts into the concept of a Fuzion feature and uses a simple intermediate language that is friendly for static analysis tools as well as for optimizing compilers.

Fuzion was influenced by many other languages including Java, Python, Eiffel, Rust, Ada, Go, Lua, Kotlin, C#, F#, Nim, Julia, Clojure, C/C++, Scala, and many more. The goal of Fuzion is to define a language that has the expressive power present in these languages and allow high-performance implementation and powerful analysis tools. Furthermore, Fuzion addresses requirements for safety-critical applications by adding support for contracts that enable formal specification and detailed control over runtime checks.

Keywords: safety, static analysis, programming language, certification

1 Introduction

Many current programming language are getting more and more overloaded with new concepts and syntax to solve particular development or performance issues. Languages like Java/C# provide classes, interfaces, methods, packages, anonymous inner classes, local variables, fields, closures, etc. And these languages are currently extended further by the introductions of records/structs, value types, etc. The possibility for nesting of these different concepts results in complexity for the developer and the tools (compilers, VMs) that process and execute the code.

For example, the possibility to access a local variable as part of the closure of a lambda expression in Java may result in the compiler allocating heap space to hold the contents of that local variable. Hence, the developer has lost control over the allocation decisions made by the compiler.

In Fuzion, the concepts of classes, interfaces, methods, packages, fields and local variables are unified in the concept of a Fuzion feature. The decision where to allocate the memory associated with a feature (on the heap, the stack or in a register) is left to the compiler just as well as the decision if dynamic type information is added or not. The developer is left with the single concept of a feature, the language implementation takes care for all the rest.

2 Fuzion Features

A Fuzion applications consists of nested feature declarations. The main operation performed on a feature is a feature call.

2.1 Components of a Feature Declaration

Feature Name A Fuzion feature is identified by a name, similar to the name of a class or a function in other languages. The name may be a plain identifier, or an operator such as *infix* + or *postfix !*, the difference between calling a named feature or applying an operator is purely syntactical.

Formal Arguments Features may have formal arguments, which are themselves features implemented as fields. On a call to a feature with formal arguments, actual arguments have to be provided to the call.

Result Type The result of a feature call is an instance of the feature. Alternatively, a feature may declare a different result type, then it must return a value of that type on a call.

Parametric Types Features may have type parameters.

Closure Features are nested, i.e., every feature is declared within the context of an outer feature. The only exception is the universe, which is the outermost feature in any Fuzion system. A feature can access features declared in its outer features. This means, a feature declaration also declares a closure of the feature and its context.

Inheritance Clause Fuzion features can inherit from one or several other features. When inheriting from an existing feature, inner features of the parent become inner features of the heir. Inherited features can be redefined. In particular, when inheriting from a feature with abstract inner features, one can implement the inherited abstract features.

Inheritance and redefinition in Fuzion does not require dynamic binding. By default, types defined by features are value types and no runtime overhead for dynamic binding or heap allocation is imposed by inheritance.

Contract A feature may declare a contract that specifies what the features does and under which conditions the feature may be called [1]. This will be handled in more detail below in the section *Design by Contract*.

Implementation Features are implemented as one of

- a routine providing code that is executed on a call,
- a field providing a memory slot that stores a value and whose contents are returned on a call,
- an abstract feature with no implementation and that cannot be called directly, but that can be redefined,
- a choice defining a union type, or
- an intrinsic implemented by the compiler or interpreter.

A feature implemented as a routine or as a choice can contain inner feature declarations.

2.2 Feature Example

Here is an example that declares a feature *point* that is similar to a struct or record in other languages:

```
point(x, y i32) is { }   # declare point as a feature with 2 args
p1 := point 3, 4         # create instance of point
```

2.3 Features as Types

A feature declaration implicitly declares a type of its instances. In the example above, the feature declaration for *point* declares the type *point* that can be used to declare a field, so we could, e.g., declare a new feature that takes an argument of type *point*:

```
draw(p point) is
  drawPixel p.x, p.y
```

Features implemented as a routine define a product type whose components are the types of the fields defined for that routine.

3 Specific Language Features

3.1 Loops

Fuzion has a powerful syntax for loops. Nevertheless, loops are syntactic sugar that is translated into feature declarations and tail recursive calls [2]. Loop index variables are automatically immutable and analysis of loop code is simplified.

3.2 Tuples

Tuples in Fuzion are provided by a generic standard library feature *Tuple*. The open list of generic parameters specifies the types of each element and their number. Here is an examples of a feature that splits a 16-bit unsigned integer *v* into two bytes returned as a tuple:

```
bytes(v u16) => ((v >> 8) & 255, v & 255)
```

The tuple can be unpacked on a call to *bytes*:

```
(hi, lo) := bytes(12345)
say "Bytes: $hi $lo"
```

Similar to the type defined for a routine, a tuple defines a product type, but without giving it an explicit name.

3.3 Choice Types

Fuzion provides choice types (also called tagged union or variant types). The simplest example of a choice type is the type *bool*, which is a choice between types *TRUE* and *FALSE*. *TRUE* and *FALSE* are themselves declared as features with no state, i.e., no fields containing any data.

Another example for a choice type from the standard library is *Option<T>*, which is a generic choice type that either holds a value of type *T* or *nil*, while *nil* is a feature with no state declared in the standard library.

A match statement can be used to distinguish the different options in a choice type, e.g.,

```
mayBeString Option<string> = someCall()
match mayBeString
  s String  => say s
  _ nil     => say "no string"
```

Together with the product types provided by tuples or routines, Fuzion provides algebraic data types.

3.4 First-class Functions

Fuzion offers first-class functions (lambdas) and maps these to Fuzion features that inherit from a standard library feature called *Function*.

4 Design by Contract

Fuzion features can be equipped with pre- and post-conditions to formally document the requirements that must be met when a feature is called and the guarantees given by a feature. An example is a feature that implements a square root function for 32-bit integers:

```
sqrt(a i32) i32
  pre
    a >= 0
  post
    # result^2 shall be less or equal toa
    result * result <= a,
    # (result+1)^2 shall be larger than a, may overflow i32
    (result + 1) > a / (result + 1),
    # result shall not positive
    result >= 0
is
  if a == 0
    0
  else
    # iteratively calculate root
    for
      # start iteration with 1
      r := 1, next
      # next in iteration is middle of r, a/r
      next := (r + a/r) / 2
    until r == next
      r
```

In this case, the function defines the pre-condition that its argument *a* is non-negative. A call of this function with a negative value will result in a runtime error. On the other hand, its post-conditions make a clear statement about the result: The result will be the largest value that, when squared, is <= a.

4.1 Checking Pre- and Post-conditions

Pre- and post-conditions can be classified for different purposes. Default qualifiers provided in the standard library are

safety protects pre-conditions that are required for the safety of an operation.

An example is the index check pre-condition of the intrinsic operation to access an element of an array: Not performing the index check would allow arbitrary memory accesses and typically would break the applications safety.

This qualifier should therefore never be disabled unless you are running code in an environment where performance is essential and safety is irrelevant.

debug is generally for debugging, it is set if debugging is enabled.

debug(n) is specific for enabling checks at a given debug level, where higher levels include more and more expensive checks.

pedantic is for conditions that a pedantic purist would require, that more relaxed hacker would prefer to do without.

analysis is used for conditions that are generally not reasonable as runtime checks, either because they are prohibitively expensive or even not at all computable in this finite universe. These conditions may, however, be very useful for formal analysis tools that do not execute the code but perform techniques such as abstract interpretation or formal deduction to reason about the code.

Runtime checks for pre- and post-conditions can be enabled or disabled for each of these qualifiers. This gives a fine-grain control over the kind of checks that are desired at runtime. Usually, one would always want to keep safety checks enabled in a system that processed data provided from the outside to avoid vulnerabilities such as buffer overflows. However, in a closed system like a controller of a launcher, it might make sense to disable checks if a runtime error would mean definite loss of the mission, while an unexpected intermediate value may still result in a useful final result of a calculation.

5 Immutability in Fuzion

Fuzion encourages the use of immutable data by simple syntax for the declaration of immutable fields. Also, the use of tail calls for loops automatically converts index variables used in that loop into immutable variables with a life span of a single loop iteration.

Since immutability is essential to ensure correctness of parallel execution within threads that do not rely on locks or similar synchronization mechanisms, Fuzion's analyzer will verify that data shared between threads is immutable.

6 Memory Management in Fuzion

Fuzion to a large extend relies on static analysis to reduce memory management overhead. Instances are by default value instances that do not require heap allocation. Furthermore, immutability in many cases avoids the need to keep a shared copy on the heap. For dynamic calls, heap allocation and dynamic binding overhead is avoided by specialization of calls.

Only for those instances for which all of these optimizations would fail, in particular instances shared between threads or long-lived instances with mutable fields, heap allocation will be required. Memory allocated on the heap will be reclaimed by a real-time garbage collector [3].

7 Fuzion Implementation

The Fuzion implementation consists of several, independent parts from a front-end performing parsing and syntax-related tasks that creates the intermediate representation (IR), via a middle-end that collects the modules needed by a Fuzion application, to the analyzers, optimizers and several backends.

7.1 Fuzion IR

Fuzion has a very simple intermediate representation. The dominant instruction is a call. The only control structure is a match operation. Loops are replaced by tail recursive calls, so there is no need in the compiler or analysis tools to handle loops as long as (tail) recursion is supported and optimized.

Clazzes in the Fuzion IR When creating the intermediate representation, Fuzion features from the source code are instantiated with actual type parameters. This are referred to as Fuzion *clazzes* (sic). Any analysis tool or compiler working on the IR hence does not need to handle parametric types.

Clazzes come in one of five flavors depending on their underlying feature:

- *Routine* – 'normal' Fuzion feature with code to execute.
- *Field* – a feature that can store data.
- *Abstract* – an abstract Fuzion feature
- *Choice* – a Fuzion feature that defines a union type.
- *Intrinsic* – a feature implemented by the compiler.

All clazzes except *choices* may be equipped with contracts.

Instructions in the Fuzion IR A clazz of type routine contains code that consists of very simple (bytecode-) instructions using a simple stack machine. There are only eight intermediate instructions:

- *Assign* – an assignment to a field
- *Box* – boxing a value to a (heap-allocated) reference type
- *Call* – a call to a clazz.
- *Current* – the current instance
- *Const* – a constant value of primitive of compound type.
- *Match* – a match instruction to determine the type of a value of a choice type
- *Tag* – create an instance of a choice type from a value whose type is one of the type parameters of the choice.
- *Pop* – remove value from the execution stack, used when call result is ignored.

The Fuzion IR can be stored in a file called a Fuzion Application (*fapp*) such that the IR can be applied to a variety of different following steps for static analysis, code generation, dynamic analysis (code coverage), optimizers, etc.

7.2 Fuzion Analyzer

Static analysis on the Fuzion IR will be used to ensure different aspects of correctness of the application. Tasks for the analyzer include

- field initialization: ensure that fields are initialized before accessed. Usually, this is ensured by the Fuzion syntax, but calls in a routine's body could result in accesses to uninitialized fields (as final fields in Java [4]).
- thread safety: ensure that fields accessed by different threads are immutable when these accesses occur to achieve thread-safety similar to Rust [5].
- module safety: for a library module to be safely usable in different context, static analysis should ensure that no external call could access undefined or intermediate state.
- ensure that contracts are satisfied. In particular, preconditions of intrinsics that are qualified with *safety* such as indexed accesses to arrays could be analyzed statically [6].
- ensure higher level of correctness defined in contracts qualified with *analysis* using quantors (using tools like KeY [7]).

Currently, the analyzers are still work on in progress, only the basic analysis required by the back ends is done.

7.3 Fuzion Optimizer

The Fuzion Optimizer modifies the intermediate representation of a Fuzion application. In particular, it determines the life spans of values to decide if they can be stack allocated or need to be heap allocated and it specializes feature implementations for the actual argument types and the actual generic arguments provided at each call.

This means that runtime overhead for heap allocation and garbage collection will be avoided as much as possible, most values can be allocated on the runtime stack. Additionally, runtime type information such as vtables will be required only in very few cases where dynamic binding cannot be avoided. An example is a data structure like a list of some reference type with elements of different actual types that are stored in this list.

7.4 Fuzion Back-Ends

Fuzion currently has two back-ends: An interpreter written in Java running on OpenJDK and a back-end creating C source code processed by gcc or clang. It is planned to add further back-ends, in particular for LLVM and Java bytecode.

Thanks to the very simple intermediate code, the back ends are relatively simple. The core of the C backend, e.g., consists of less than 1000 lines of well documented Java code.

7.5 Certification Artifacts

Due to the simple structure of the backend it is expected that certification of generated code will be simplified. The IR removes syntactic sugar, but should remain simple enough to allow for manual analysis and creation of certification artifacts. Furthermore, it can be expected that qualification of a backend for use in safety-critical systems will be simpler compared to other languages.

8 Conclusion / Next Steps

The Fuzion language definition and implementation is still in an early stage. A first version of the language and its implementation was presented at FOSDEM 2021 [8]. The main areas of ongoing work are the definition of a standard library, interfaces to other languages, IDE integration and documentation. I expect the presented approach to have a huge potential, in particular in the safety-critical domain. The combination of a powerful language with a simple core that is open for static analysis tools can improve the software development productivity, the safety of the resulting software as well as the performance.

9 Availability

Fuzion is available as source code on github [9]. The Fuzion portal website *flang.dev* [10] provides an interactive tutorial with many examples, a tutorial and many background documents on the design of Fuzion.

References

[1] B. Meyer, "Applying "design by contract"," *Computer*, vol. 25, p. 40–51, Oct. 1992.

[2] W. D. Clinger, "Proper tail recursion and space efficiency," *SIGPLAN Not.*, vol. 33, p. 174–185, May 1998.

[3] F. Siebert, "Concurrent, parallel, real-time garbage-collection," in *Proceedings of the 2010 International Symposium on Memory Management*, ISMM '10, (New York, NY, USA), p. 11–20, Association for Computing Machinery, 2010.

[4] "Use of uninitialized final field - with/without 'this.' qualifier." `https://stackoverflow.com/questions/13864464/`, 2012.

[5] S. Klabnik and C. Nichols, *The Rust Programming Language: Ch. 16 Fearless Concurrenty*. USA: No Starch Press, 2018.

[6] R. Rugina and M. C. Rinard, "Symbolic bounds analysis of pointers, array indices, and accessed memory regions," *ACM Trans. Program. Lang. Syst.*, vol. 27, p. 185–235, Mar. 2005.

[7] B. Beckert, R. Hähnle, and P. H. Schmitt, *Verification of Object-Oriented Software: The KeY Approach*. Berlin, Heidelberg: Springer-Verlag, 2007.

[8] "FOSDEM." `https://fosdem.org`, 2021.

[9] "Fuzion Souces." `https://github.com/fridis/fuzion`.

[10] "Fuzion Portal." `https://flang.dev`, 2021.

First Steps Towards an IEEE 802.1AS Clock for EDF Scheduling in Distributed Real-Time systems

Héctor Pérez Tijero, Diego García Prieto, and J. Javier Gutiérrez
Universidad de Cantabria, 39005-Santander, SPAIN; Email: {perezh,gprietod,gutierjj}@unican.es

Abstract

The EDF (Earliest Deadline First) scheduling policy can take advantage of the existence of a synchronization clock mechanism to reach higher utilizations in distributed real-time systems, as indicated by theoretical research. In this work, we give the first steps to enable the usage of EDF and a global clock in a distributed system. To this end, our proposal relies on Linux to provide the global clock and on a Real-Time Operating System (RTOS) to provide the global EDF policy. The objective is the evaluation of global EDF to verify to what extent the theoretical results are met.

Keywords: EDF, clock synchronization, real-time systems, distributed systems.

1 Introduction

EDF scheduling can get a better usage of the processor in real-time single-processor systems compared to fixed-priority (FP) scheduling; moreover, FP does not always have the commonly attributed advantages over EDF [1]. In the context of real-time distributed systems and EDF, the work in [2] shows how the schedulability of a system can be enhanced through the availability of a global clock (i.e., a clock synchronization mechanism) by using a proper scheduling deadline assignment, when tasks are composed of a sequence of sub-tasks with precedence relations which may be executed in different processors.

According to [2], global EDF scheduling can be applied as long as each node belonging to the distributed system has access to a global clock (i.e., a clock which is synchronized using the same timing source). To this end, the IEEE 802.1AS specification [3] proposes a protocol for the transport of timing over bridged local area networks. This specification is part of the Time-Sensitive Networking (TSN) set of standards [6], an ongoing effort to enable deterministic data transfer in Ethernet networks.

Implementations for the clock synchronization mechanism proposed by IEEE 802.1AS are available for different operating systems, including Linux. Results from a recent survey to 120 industry practitioners in the field of real-time embedded systems [4] show that Linux is present in 55.88% of respondents' applications, from which 42,2% also use RTOS (Real-Time Operating Systems). For instance, running real-time applications over Linux has been considered for space or HPC applications [5]. However, the design of the Linux kernel favors throughput over determinism and therefore it should not be used with its default configuration for real-time systems. Some of the most notable approaches to bound response times of real-time applications in Linux include the PREEMPT_RT kernel patch and CPU isolation capabilities [5]. While the former aims at reducing the latency by making most of the kernel preemptable, the latter aims at isolating a processing core for one or more tasks or sub-tasks.

Linux provides an implementation of EDF scheduling combined with the CBS (Constant Bandwidth Server) algorithm [7]. Under this scheduling policy, the current scheduling deadline of a sub-task is initialized to the current time plus a configured period set by CBS, where each sub-task is associated with a budget and a period. Thus, this policy does not fit well with the concept of global EDF [2], where scheduling deadlines of each sub-task are referenced to the release time of the task (first sub-task), possibly in a different processor.

On the other hand, MaRTE OS [8] is a real-time kernel mainly written in Ada [9], which follows the POSIX.13 minimal real-time system. This kernel implements most of the services defined by the Real-Time Systems annex [9], including the *EDF_Across_Priorities* dispatching policy. The implementation of the EDF policy in MaRTE OS allows programming applications using global EDF as long as a global clock is available. Furthermore, MaRTE OS allows the execution of applications not only over bare machine but also over the Linux kernel with some limitations.

This paper will explore the use of a global clock in a distributed system as a foundation step for supporting global EDF scheduling. To this end, it proposes an architecture which enables clock synchronization through the facilities implemented by the Linux kernel, and scheduling services through MaRTE OS, thus making this architecture appropriate for soft real-time applications.

The paper is organized as follows. Section 2 gives some details on the proposed architecture based on MaRTE OS over Linux. Section 3 describes the configuration and changes required to integrate all the technologies used. Some performance metrics are presented in Section 4. Finally, Section 5 outlines the conclusions and our future work.

2 System architecture

This section aims to define a system architecture upon which support for global EDF scheduling could be built. Figure 1 shows a multicore system following the proposed software

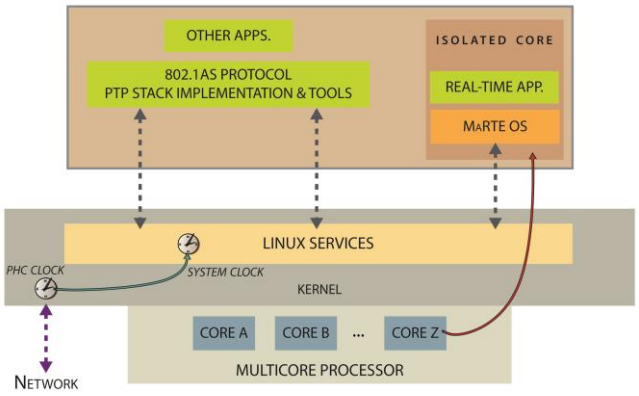


Figure 1: System architecture

architecture for integrating a global clock in distributed systems with real-time requirements.

As can be seen in Figure 1, the Linux kernel provides different services to applications, such as timing or networking. Real-time applications are executed on top of MaRTE OS, which is configured with the *linux_lib* architecture. Under this architecture, a MaRTE OS application can be executed as a standard Linux process in which concurrency and scheduling facilities are provided by MaRTE OS. It is worth noting that these applications are standard processes and therefore may be interrupted by Linux kernel activities. To enhance the determinism of a MaRTE OS application executed on top of Linux, our proposal applies the isolation capabilities provided by the Linux kernel.

Isolating a CPU can enhance the predictability of applications by limiting user and kernel preemptions. The Linux kernel provides several features to facilitate the execution of threads in isolated CPUs. However, these features do not provide full isolation and they do not consider some sources of interferences such as shared caches, global work queues or interprocessor interrupts. Furthermore, there are also a few kernel threads bound to each CPU whose workload cannot be migrated.

The concept of global time is provided by means of the implementation of the IEEE 802.1AS protocol, which is responsible for synchronizing the PHC clock (Ptp Hardware Clock) for each node belonging to the distributed system. As can be seen in Figure 1, the system clock could also be synchronized in order to facilitate access to the global time from user-space applications. Finally, Figure 2 shows a possible configuration for the synchronization of PHC and system clocks in IEEE 802.1AS systems. In this example, the timing source is located in the switch but other configurations are allowed by the standard.

3 The distributed real-time platform

A prototype of the architecture introduced in Section 2 has been implemented as a proof of concept in order to validate the integration of the required software and provide preliminary performance metrics.

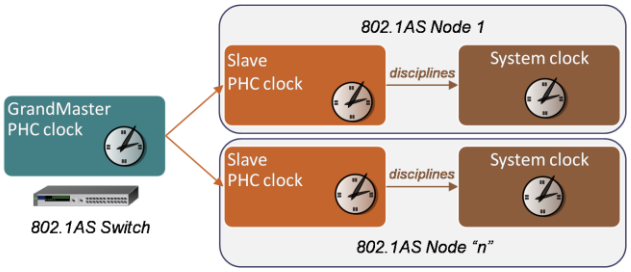


Figure 2: Example of clock synchronization in 802.1AS system

To this end, the *linux_lib* architecture of MaRTE OS has been modified to provide access to the global clock from the application level. To enable this feature, the timing functions located in the Hardware Abstraction Layer have been modified to use the clock with synchronized time provided by the PTP stack. On the other hand, the configuration of the Linux kernel requires tuning both compilation and runtime parameters, which are briefly described next.

CPU isolation at runtime is provided by means of *isolcpus* and *cpuset* facilities [10]. These facilities allow one or more CPUs to be isolated (i.e., unavailable to the scheduler) and therefore the execution of any MaRTE OS process in an isolated CPU does not compete with the rest of the workload.

To enhance the isolation, the CONFIG_NO_HZ_FULL kernel parameter [10] has been enabled. This parameter allows a CPU to run in *adaptative* tick mode (i.e., the kernel tick is offloaded to another CPU when the isolated CPU is idle or running a single thread). However, this parameter cannot remove a residual 1 Hz tick which still remains as a source of interference. Additionally, the isolated CPU is switched off during boot time to force the kernel to assign the regular workload to the non-isolated cores.

The PREEMPT_RT patch makes several changes to the kernel to favor determinism over throughput. One of them is converting handler interrupts into preemptible kernel threads. This change may compromise the benefits of *adaptative* tick mode, as it requires at most one single runnable thread in the CPU. Since some kernel threads are allocated *per CPU* and cannot be migrated to another CPU, several runnable threads can be available at the same time, thus enabling the kernel tick. Furthermore, the design of the *linux_lib* architecture of MaRTE OS relies on POSIX signals for the communication with the Linux kernel, which increases the complexity of the system configuration when they are handled by kernel threads.

As a result, the proposed architecture relies instead on the low latency profile for the kernel, which integrates some of the changes proposed by the PREEMPT_RT patch but without compromising CPU isolation. As a side note, the use of MaRTE OS as a threading library on top of Linux allows the execution of a multithreaded application in *adaptative* tick mode since there is only one runnable thread from the perspective of the Linux kernel.

Finally, LinuxPTP[7] is a suite of tools for Linux that enable the configuration and access to a global clock in a distributed

[7] Available at http://linuxptp.sourceforge.net/

system. For our purposes, two tools from the toolset are particularly relevant: (1) *ptp4l* to synchronize the PHC clocks through the network, and (2) *phc2sys* to synchronize two local clocks in each node. Besides following the IEEE 802.1AS specification, both tools are configured to provide monotonically increasing times (i.e., clock corrections are performed by changing the clock frequency), as it is more suitable for real-time applications.

4 Experimental measurements

This section aims to obtain preliminary performance metrics and assess the synchronization capabilities of using a global clock in the proposed architecture.

The hardware platform consists of two quad-core 1.9 GHz nodes connected through an isolated 1 Gbps Ethernet network. Both nodes and the switch are compliant with the IEEE 802.1AS standard and run the ptp stack to implement the synchronization of clocks according to the configuration illustrated in Figure 2.

The implementation of the architecture depicted in Figure 1 has used the following software components: (1) Linux kernel v4.4.256-rt214 compiled using the parameters described in Section 3; (2) MaRTE OS v2017 using the *linux_lib* architecture, together with the required changes for accessing to the global clock described in Section 3; (3) *ptp4l* v3.0 to implement the PTP stack; and (4) *phc2sys* v3.0 to synchronize system and PHC clocks.

To better compare the metrics obtained, two different scenarios are defined: (1) the *MaRTE OS over Linux* scenario, which represents the proposed approach; and (2) the *Reference* scenario in which the test applications use the same features as the previous scenario (e.g., CPU isolation) but they execute directly on top of the Linux kernel.

4.1 Overhead metrics

The first case study aims to estimate the overhead associated with the reading of the global clock. This operation is executed 10,000,000 times, and the average, maximum, and minimum times are estimated, together with the standard deviation.

Table 1 shows the metrics obtained for both scenarios under evaluation, which obtain a high maximum time compared to the average and minimum values. This difference may be caused by interferences of the Linux kernel whose design is not specifically oriented to real-time systems. According to the specialized literature [11], this can lead to worst-case latencies in the range of tens of μs.

Scenario	Max	Avg	Min	Std Dev
Reference	27.30	0.07	0.07	0.01
MaRTE OS over Linux	38.80	3.40	3.40	0.07

Table 1: Overhead in accessing the global clock (time in μs)

It is also worth noting that *MaRTE OS over Linux* scenario obtains higher average values. In Linux, the *clock_gettime* function is usually supported as vDSO (virtual Dynamic Shared Object) to improve its performance [10]. However, the *clock_gettime* call in the *linux_lib* architecture of MaRTE OS is internally redirected to the Linux kernel through an explicit system call, thus adding an extra overhead in relation to the native case. Finally, the results show that the deviation standard is below 0.1 μs in both scenarios.

4.2 Event-handling latency metrics

The second case study aims to estimate the latency associated with the handling of an external event. In particular, this test measures the delay in the response to a periodic timer event (i.e., it executes *clock_nanosleep* followed by *clock_gettime* using the global clock). In this case, the requested time through *clock_nanosleep* should be nearly identical to the time measured by *clock_gettime*. This evaluation is executed 1,000,000 times with a period of 100μs, and the average, maximum, and minimum times are estimated, together with the standard deviation.

Table 2 shows the metrics obtained for this case study. Again, the measurements obtained for *MaRTE OS over Linux* are higher on average. This is due to the design of MaRTE OS, which relies on POSIX signals to communicate with the Linux kernel. However, the results show that this overhead is lower than the maximum interferences caused by the Linux kernel in both scenarios. Finally, the deviation standard remains below 1μs in both scenarios, although it is slightly lower for *MaRTE OS over Linux*. Therefore, the proposed approach adds some extra overhead, but it also reduces the dispersion.

Scenario	Max	Avg	Min	Std Dev
Reference	30.20	6.50	5.10	0.60
MaRTE OS over Linux	23.70	18.60	16.20	0.32

Table 2: Event-handling latency for a timer event (time in μs)

4.3 Timing synchronization metrics

The last case study aims to evaluate the synchronization of the global clock in a distributed system. To this end, two nodes will periodically raise a digital signal using the global clock as the timing source, and starting at the same time. Then, an oscilloscope will measure the delay between the two digital signals. This operation is executed 50,000 times, and the average and maximum times together with the standard deviation are estimated. Figure 3 and Figure 4 show the results obtained for both scenarios using a period of 100μs. Note that negative values just indicate that the second signal is raised first.

Both scenarios obtain similar maximum absolute delays which are below 10μs. Furthermore, the standard deviation obtained by each scenario is in the same range as the second case study.

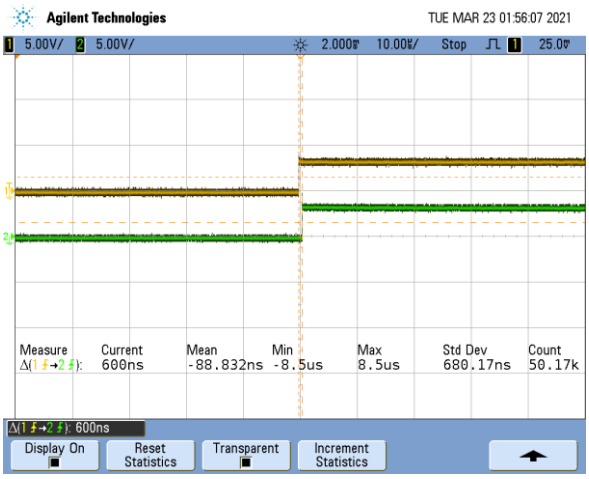


Figure 3: Metrics for clock synchronization in the Reference case

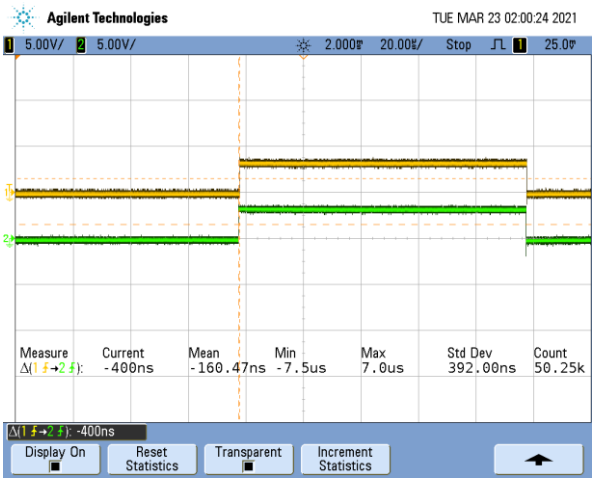


Figure 4: Metrics for clock synchronization in *MaRTE over Linux*

5 Conclusions and future work

The use of a globally synchronized clock in a distributed system could enable the use of EDF scheduling based on end-to-end global deadlines. However, support for EDF scheduling policy is not widely adopted in current operating systems.

MaRTE OS provides support for EDF scheduling as described in the Real-Time Annex of Ada. However, support for PTP stack is not available, so this paper proposes an architecture based on Linux for providing global and local timing services but using MaRTE for threading and scheduling services in an isolated core. The proposed approach does not provide guarantees about full isolation, as the 1 HZ tick still remains present in the isolated core. Unlike the native scenario, *MaRTE OS over Linux* also enables the execution of a multithreading application in the isolated core.

The results obtained show that the proposed approach could be of interest for applications with soft real-time requirements whose deadlines are in the range of hundreds of microseconds. Finally, future work includes the evaluation of global EDF scheduling in the proposed platform.

Acknowledgements

This work was partially supported by the Spanish Government and FEDER funds (AEI/FEDER, UE) under grant TIN2017-86520-C3-3-R (PRECON-I4).

References

[1] G. Buttazzo, “Rate Monotonic vs. EDF: Judgment Day”, *Real-Time Systems 29(1)*, pp. 5-26 (2005).

[2] J. M. Rivas, J. J. Gutiérrez, J. C. Palencia, and M. González Harbour, “Deadline Assignment in EDF Schedulers for Real-Time Distributed Systems”, *IEEE Transactions on Parallel and Distributed Systems 26(10)*, pp. 2671-2684 (2015).

[3] IEEE 802.1AS - *Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, 2020.

[4] B. Akesson, M. Nasri, G.Nelissen, S. Altmeyer and R. I. Davis, “An Empirical Survey-based Study into Industry Practice in Real-time Systems” in *Proceedings of the 41st Real-Time Systems Symposium* (RTSS 2020), pp. 3-11, 2020.

[5] M. Madden, “Challenges Using Linux as a Real-Time Operating System”, AIAA SciTech Forum (Software Challenges in Aerospace), doi:10.2514/6.2019-0502, 2019.

[6] IEEE 802.1 Time-Sensitive Networking (TSN) Task Group. Available online: https://1.ieee802.org/tsn/ [Retrieved March, 2021]

[7] L. Abeni and G. Buttazzo, “Integrating multimedia applications in hard real-time systems” in *Proceedings of the 19th IEEE Real-Time Systems Symposium*, Madrid, pp.4-13, 1998.

[8] M. Aldea and M. González, “MaRTE OS: An Ada Kernel for Real-Time Embedded Applications” in Proc. of the International Conference on Reliable Software Technologies, Ada-Europe 2001, Leuven, Belgium, in *Lecture Notes in Computer Science*, LNCS 2043, 2001.

[9] Ada 2012 Ref. Manual. Language and Standard Libraries - International Standard ISO/IEC 8652/2012 (E). doi: 10.1007/978-3-642-45419-6 (2013).

[10] The Linux Kernel documentation. Available online: https://www.kernel.org/doc/Documentation/ [Retrieved March, 2021].

[11] F. Reghenzani, G. Massari, and W. Fornaciari, “The Real-Time Linux Kernel: A Survey on PREEMPT_RT”, *ACM Comput. Surv. 52*, 1, Article 18, 36 pages. doi: 10.1145/3297714, 2019.

Pyramids, Reduction Expressions, and Final Puzzles

John Barnes

11 Albert Road, Caversham, Reading, RG4 7AN, UK; Tel: +44 118 9474125; email: john@jbinformatics.co.uk

Hello readers

This is going to be the last of these notes. My goal was to cover the puzzles that I have found sufficiently interesting to be worthy of mention at Ada-Europe conferences. This year the conference was electronic and my Avatar and I gave a bit of a talk and ended by mentioning four puzzles.

But first I must give the solution to the piles of cannonballs problem. Recall that soldiers were asked to pile their cannonballs into square pyramids such that the number of balls in each pyramid is itself a square number. Each layer of a square pyramid comprises a square number of balls. The first few pyramidal numbers are 1, 5, 14, 30, and 55. Note that 1 is of course a square number so the first pyramidal number that is also a square is simply 1. The question is what is the second pyramidal number that is also a square number?

The answer is $4900 = 70^2$.

The pyramidal numbers are easily computed by

```
Pn := 0;
for J in 1 .. N loop
   Pn := Pn + J**2;
end loop;
```

or using the new reduction expression of Ada 2022 by

```
Pn := ([for J in 1 .. N => J**2]'Reduce("+", 0));
```

and taking N = 24 we find that $P_{24} = 4900$.

Strangely, that is it. There are no more pyramidal numbers that are also squares.

The four puzzles that were mentioned at the conference this year are all taken from the eleventh book of Problematical Recreations from Litton Industries of Beverly Hills. It was given to me by Angela Fox Dunn who is the author of a book entitled Mathematical Bafflers and was originally published by McGraw-Hill but is now available from Dover.

1. If the equal sides of an isosceles triangle are given (say x), what length of the third side will provide maximum area?

2. By the time the radius of a certain pearl has increased 1mm, the area will have increased as much (in mm squared) as the volume (in mm cubed). If the pearl is an exact sphere, what is its radius now?

3. A pencil, eraser, and notebook together cost $1.00. A notebook costs more than two pencils, and three pencils cost more than four erasers. If three erasers cost more than a notebook, how much does each cost?

4. If THAT = (AH)×(HA) what is THAT?

The basic answers are as follows:

1. $\sqrt{2} \times x$

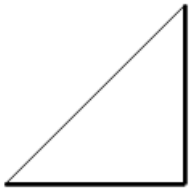
2. $1/2 + \sqrt{33} / 6 = 1.457...$

3. Pencil = 26, Eraser = 19, Notebook = 55.

4. THAT = 6786

In more detail:

1. The trick is to turn the triangle on its side, thus



since area is half base times height and height will be largest when it is vertical, it is a right-angled triangle and the other side is therefore $\sqrt{2} \times x$.

2. We have $4\pi(r+1)^2 - 4\pi r^2 = (4\pi(r+1)^3 - 4\pi r^3)/3$ from which $3r^2 - 3r - 2 = 0$. Hence r as above.

3. The inequalities can be written as $9e > 3n > 6p > 8e$. We now combine to get limits on e.

Now $n < 3e$, $p < 3/2e$. So $p+n+e = 100 < (3/2 + 3 + 1)e = 51/2e$. So $e > 18$.

And $n > 8/3e$, $p > 4/3e$. So $p+n+e = 100 > (4/3 + 8/3 + 1) e = 5e$. So $e < 20$.

Hence e =19. We then find that $p > 4/3e$, so $p > 25$ and $p < 3/2e$, so $p < 29$.

Try p = 28, implies n = 53. No. n must be greater than 2p.

Try p = 27, implies n = 54. No. But nearly.

Try p = 26, implies n = 55. OK.

This can be neatly explained using triangular graph paper.

4. Since (H+10A)×(A+10H) = 1001T+100H+10A we get 1001T = (A+10H)×(H+10A–10). The factors of 1001 are 7, 11, and 13. Only T = 6 gives the integral solution that 6006 = 78 × 77. Hence THAT = 6786.

I must leave you with one other problem that I find amusing and you can try out on any passing friend.

Consider (I)×(AM)×(NOT) = (SURE).

The reason why I am not sure is that there are two solutions, namely 1×26×345 = 8970 and 2×14×307 = 8596.

Take care!!

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o KU Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Hubert B. Keller
Karlsruher Institut für Technologie (KIT)
Institut für Angewandte Informatik (IAI)
Campus Nord, Gebäude 445, Raum 243
Postfach 3640
76021 Karlsruhe
Germany
Email: Hubert.Keller@kit.edu
URL: ada-deutschland.de

Ada-France

attn: J-P Rosen
115, avenue du Maine
75014 Paris
France
URL: www.ada-france.org

Ada-Spain

attn. Sergio Sáez
DISCA-ETSINF-Edificio 1G
Universitat Politècnica de València
Camino de Vera s/n
E46022 Valencia
Spain
Phone: +34-963-877-007, Ext. 75741
Email: ssaez@disca.upv.es
URL: www.adaspain.org

Ada-Switzerland

c/o Ahlan Marriott
Altweg 5
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: president@ada-switzerland.ch
URL: www.ada-switzerland.ch

Ada-Europe Sponsors

27 Rue Rasson
B-1030 Brussels, Belgium
Contact:Ludovic Brenta
ludovic@ludovic-brenta.org

46 Rue d'Amsterdam
F-75009 Paris, France
Contact: Jamie Ayre
sales@adacore.com
www.adacore.com

2 Rue Docteur Lombard
92441 Issy-les-Moulineaux Cedex
France
Contact: Jean-Pierre Rosen
rosen@adalog.fr
www.adalog.fr/en/

22 St. Lawrence Street
Southgate, Bath BA1 1AN
United Kingdom
www.capgemini.com

4545 E. Shea Blvd. #210
Phoenix, AZ 85028
USA
Contact: Laurent Meilleur
sales@ddci.com
www.ddci.com

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

In der Reiss 5
D-79232 March-Buchheim
Germany
Contact: Frank Piron
info@konad.de
www.konad.de

3271 Valley Centre Drive,
Suite 300
San Diego, CA 92069, USA
Contact: Shawn Fanning
sfanning@ptc.com
www.ptc.com/developer-tools

Signal Business Centre
2 Innotec Drive, Bangor
North Down BT19 7PD
Northern Ireland, UK
enquiries@sysada.co.uk
www.sysada.co.uk

1090 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

Corso Sempione 68
20154 Milano
Italy
Contact: Massimo Bombino
massimo.bombino@vector.com
www.vector.com

Beckengässchen 1
8200 Schaffhausen
Switzerland
Contact: Ahlan Marriott
admin@white-elephant.ch
www.white-elephant.ch

United Kingdom
Contact: Chris Nettleton
nettelton@xgc.com
www.xgc.com


Ada
europe