The journal for the international Ada community

Ada User Journal

Volume 41
Number 2
June 2020

Produced by Ada-Europe

Editor in Chief

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

Ada User Journal Editorial Board

Luís Miguel Pinho *Associate Editor*	Polytechnic Institute of Porto, Portugal *lmp@isep.ipp.pt*
Jorge Real *Deputy Editor*	Universitat Politècnica de València, Spain *jorge@disca.upv.es*
Patricia López Martínez *Assistant Editor*	Universidad de Cantabria, Spain *lopezpa@unican.es*
Kristoffer N. Gregertsen *Assistant Editor*	SINTEF, Norway *kristoffer.gregertsen@sintef.no*
Dirk Craeynest *Events Editor*	KU Leuven, Belgium *Dirk.Craeynest@cs.kuleuven.be*
Alejandro R. Mosteo *News Editor*	Centro Universitario de la Defensa, Zaragoza, Spain *amosteo@unizar.es*

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 41
Number 2
June 2020

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

I cannot find another way to start this editorial other than apologising for the late production of this issue, as it already happened with the previous one. The COVID-19 pandemic has been affecting not only the organization of events and everything that used to imply our physical presence, but also the preparation of the Ada User Journal. Due to this delay, the least we can do is to make the issue immediately available on the journal web page, in digital format. Naturally, the printed copies will be sent to our subscribers as soon as possible.

Concerning the contents, we are happy to include, for the first time, a paper derived from a presentation in the Ada Developer Room at FOSDEM. FOSDEM is an event that aims at promoting the widespread use of Free and Open Source software, which is being organised every year since 2001, in Brussels, Belgium. This international event hosts many activities that take place in rooms spread across several buildings of the Solbosch campus of the ULB (Université Libre de Bruxelles). One of these rooms is dedicated to the Ada language. The paper presents the Spunky project, developed by Martin Stein, from Genode Labs, Germany. The idea of this project is to provide a microkernel for the Genode OS framework, which is a free open-source tool kit for building highly secure component-based operating systems. We hope that by publishing this paper in our pages, participants in future editions of the Ada Developer Room at FOSDEM will be motivated to also prepare papers describing their work, sharing them with the Ada community through the AUJ.

In this June issue we also include three papers that were submitted to the Industrial Track of the 25th Ada-Europe International Conference (AEiC 2020), which had to be cancelled this year and postponed to June 2021 (further information can be found in the forthcoming events section). Authors of submitted papers were nevertheless invited to prepare contributions to the AUJ. The first paper, authored by Tewodros A. Beyene and Christian Herrera, from fortiss, Germany, presents a two-step approach for Ada code review, which combines checks of run-time errors with a more powerful formal analysis using the AdaHorn model checker. Then, the second paper introduces De-RISC, a Dependable Real-time Infrastructure for Safety-critical Computer Systems. The group of authors, from fentISS (Spain), Cobham Gaisler (Sweden), Thales (France) and BSC (Spain), describes De-RISC as a HW/SW platform meeting five fundamental requirements of safety and mission-critical applications in the space domain. They expect that De-RISC will reach commercial maturity in 2022. Finally, the third paper, from researchers of the Polytechnic University of Valencia, Spain, of fentISS, Spain, and of Thales, France, proposes an architecture for the design and deployment of real-time partitioned systems, named Time4PS.

In addition to the technical contributions included in the issue, we provide a new puzzle prepared by John Barnes, this time about the problem of the Greek Cross. In fact, we expect the puzzle page to become a permanent section of the AUJ, providing problems that will be, tentatively, somehow related to Ada. The respective solutions will be provided in the following issue, which means that in this issue John Barnes provides the solution for the problem proposed in the March issue. Please feel free to send us your solution for the Greek Cross problem – maybe it will surprise John Barnes!

Last but not the least, the reader will find the Quarterly News Digest section, with a selection of news prepared by Alejandro R. Mosteo for the period between April and June 2020, and the Conference Calendar section, prepared by its editor, Dirk Craeynest.

Antonio Casimiro
Lisboa
June 2020
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,

In this new section opening the News Digest I would like to highlight a few topics in the current number that I found personally more interesting, or out of the ordinary, or particularly relevant for some reason.

On this occasion, compilers are the stars of the show. PTC has announced the release of ObjectAda 10.2, which becomes the second compiler with full Ada 2012 support [1]. Meanwhile, the announcement of a new GNAT Community Edition is always exciting [2], but this year we are also witnessing the fast progress of the HAC compiler. The topic on future directions and blurring the lines between compilation and scripting [3] puts the spot on the many possibilities that this new open source Ada compiler may bring to the table.

In a related vein, the prototype for Jupyter notebooks with Ada posted to the Telegram Ada group [4] shows the potential of script-like capabilities for the use of Ada in learning/presentation contexts. Incidentally, this Telegram topic is the first one to appear from that source in the News Digest.

Finally, Jeremy Grosser has put together a searchable (and cloneable) archive of all posts to comp.lang.ada since 1982, with synchronization moving forward [5].

Sincerely,

Alejandro R. Mosteo.

[1] "PTC ObjectAda 10.2", in Ada-related Products.

[2] "AdaCore GNAT Community Edition 2020", in Ada-related Tools.

[3] "Script-like Jobs in Ada (Ideas for HAC)", in Ada Practice.

[4] "Jupyter Kernel for Ada", in Ada and Education.

[5] "Searchable comp.lang.ada Archive Since 1982", in Ada-related Resources.

Ada-related Events

Adalog Webinar on Formal Methods with Ada and Spark

[Past event, for the record. --arm]

From: "J-P. Rosen" <rosen@adalog.fr>
Subject: [Ann] Adalog Webinar on formal methods with Ada and SPARK
Date: Tue, 5 May 2020 12:16:22 +0200
Newsgroups: comp.lang.ada

Adalog is pleased to announce a training session as a 3-day webinar, May 27th to 29th, about using formal methods with Ada and SPARK.

This webinar is organized in cooperation with Ran Ettinger, a specialist in formal methods and professor at Ben-Gurion University and Academic College of Tel Aviv.

It will be given in French/English

All details (in French) available from:

https://adalog.fr/fr/formation_adaspark.html

Request for WG 9 Participation and Ada 202x Draft Standard Review

[The deadline for contributions is closed, but attaining membership for future occasions is timeless. --arm]

From: Pat Rogers <rogers@adacore.com>
Subject: Request for WG 9 participation and Ada 202x draft standard review
Date: Fri, 22 May 2020 13:43:22 -0700
Newsgroups: comp.lang.ada

To Whom It May Concern,

On behalf of ISO JTC 1/SC 22/WG 9, the working group responsible for the ISO Ada standard, I am writing to make two very important requests regarding the Ada language. Please forward this message to whomever you think is most appropriate within your organization if you are not that person. For that matter, if you know someone outside your organization please feel free to forward this note to them.

My first request is that you consider joining WG 9 as an official member (unless you are already a member, of course). WG 9 is responsible for the technical content and direction of the evolution of the ISO Ada standard. Your inputs would directly affect both.

Participation involves attending two meetings per year, either in person or remotely (virtually). One meeting is held in the United States, usually in the Fall, and one in Europe, immediately after the annual Ada Europe conference each Summer. Activities include reviewing and voting on the content of the ARG language proposals, setting the direction for the language as it evolves, and anything pertinent to your specific interests and backgrounds, such as creating technical reports, liaising with other Working Groups for others languages and reports (e.g., the report on language vulnerabilities), and so on.

Participation requires ISO membership. Membership in ISO committees and working groups is organized in terms of national bodies, managed by entities specific to each country. Your membership in WG 9, therefore, would entail whatever the managing entity for your specific country requires.

For example, in the United States, ISO participation is managed by INCITS, the InterNational Committee for Information Technology Standards (http://www.incits.org/). Membership in INCITS is organized in terms of individual corporations, with individuals from within those companies designated as representatives to ISO committees.

INCITS charges an annual membership fee per company. Other countries have other requirements.

I will be happy to help facilitate your membership in any way I can.

Please feel free to ask.

My second request, again on behalf of WG 9, is for your comments on the Ada 202x draft standard.

As you may know, the next revision, known informally as Ada 202x, is expected to be completed in the next several months. There are a number of important additions to the language, and your expertise in reviewing these proposals would be invaluable.

If you can, please send your written comments by 7 June, 2020, which will give us time prior to the next WG 9 meeting to analyze them.

You can use the following links to access the draft standard and the AIs.

The Annotated Ada 202X Reference Manual:

http://www.ada-auth.org/standards/2xaarm/html/AA-TOC.html

The Ada 202X Reference Manual, absent the annotations:

http://www.ada-auth.org/standards/2xrm/html/RM-TOC.html

The list of AIs: http://ada-auth.org/AI12-VOTING.HTML

For convenience, the Annotated Ada 2012 reference manual:

http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-TOC.html

Thank you in advance for your consideration.

Best regards,

Patrick Rogers

WG 9 Convenor

rogers@adacore.com

Call for Presentations: ACM HILT 2020 Workshop at SPLASH 2020.

[Due date, found below, is September 4th. --arm]

From: ric.wai88@gmail.com
Subject: Call For Presentations: ACM HILT 2020 (High Integrity Language Technologies) workshop at SPLASH 2020.
Date: Tue, 16 Jun 2020 08:17:24 -0700
Newsgroups: comp.lang.ada

This is the 6th HILT workshop, and will focus on the growing importance of large-scale, highly parallel, distributed and/or cloud applications.

The workshop will be part of SPLASH 2020 Conference, on November 15-20. The conference is tentatively planned to take place in Chicago, but may be hosted virtually, or a combination thereof, depending on the evolution of the COVID-19 pandemic.

We are currently accepting proposals for presentations, due by September 4th. The workshop program committee will select presentations and organize them into sessions.

Attendees wishing to present at the workshop should prepare extended abstracts (approx. 2-4 pages) for the proposed presentations. Full papers (6-8 pages) are also acceptable.

Key areas of interest include:

- Safe and Productive Languages and Frameworks for the development of structured parallel and/or distributed applications (e.g. Rust, Concurrent Collections, Ada 202X, Parsl)
- Broadly available technologies to support large dataset analysis and machine learning workloads (e.g. TensorFlow, Apache Spark)
- Practical tools for applying static analysis and formal methods to parallel and/or distributed/cloud applications (e.g. SPARKProver, Java Pathfinder)
- Underlying Portability Frameworks to support higher level capabilities (e.g. OpenMP, OpenACC, OpenCL, MPI)
- Key technologies to bring high-performance computing to more traditional programming environments (e.g. advanced IRs supporting parallelism and heterogeneity such as MLIR and Tapir/LLVM)

Please visit the landing page at https://2020.splashcon.org/home/hilt-2020 for more information!

Ada and Education

Strategies for Teaching Ada

[A recent book by Andrew T. Shvets -- "Beginning Ada Programming" -- elicits debate on how to best introduce Ada to beginners. This thread collects a number of responses from the author to previous criticism and the ensuing discussion. --arm]

From: Andrew Shvets <andrew.shvets@gmail.com>
Subject: Re: Beginning Ada Programming, by Andrew T. Shvets (2020)
Date: Thu, 16 Apr 2020 20:58:56 -0700
Newsgroups: comp.lang.ada

> This is not out yet, but it looks interesting and is due at the end of the month:

> https://www.springer.com/us/book/9781484254271

>

> Is the author the same Andrew Shvets who posts here sometimes?

>

> Andrew if you're here, what does the book cover in terms of e.g. new Ada features, SPARK, etc.?

>

> It's great that new books are coming out about Ada.

I just stumbled across this thread (I haven't been here a while). Hence the very late reply.

The book is pretty much the same as my "Introduction to Ada Programming". The one key advantage in the Apress version is that the Index is much much much better now. Also, I feel like the formatting is vastly improved than what it was before.

Lastly, by going with Apress, it improves the status and visibility of Ada. It's no longer the language of someone that programmed when Reagan was president. It gives it a new feel of vitality and drives home the point that Ada is not "dead".

I'm under contract with Apress for a 2nd edition for this book. That one will have much more information (such as a chapter on making GUIs). I'm still working on that one.

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Thu, 16 Apr 2020 21:04:24 -0700

> > Is the author the same Andrew Shvets who posts here sometimes?

>

> Yes, he is. I had a little conversation on the very first edition with him. I esp. objected that he started his examples with using Integer rather than user-defined types (which IMHO is the very heart of Ada). I do not know whether he changed this.

You have to understand something from the perspective of a newbie. If you're new to Ada, heard all the wonderful things about it and then get started... Most of the time you get stuck in a long chapter about types that go down the rabbit hole of explaining the entire type system of Ada right away.

Don't get me wrong, this is a core strength of Ada and it's absolutely awesome. However, when you're just starting out, it can be daunting, frustrating and discouraging. It was for me.

My personal take on this was to ease the reader into the subject. I wanted to give an overhead of how powerful types are in Ada, but not dump the reader at the deep end of the pool. As a result, I had a chapter on the basic types and then moved on to control structures, methods, OOP, etc. Later on, I went back and revisited this topic.

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Thu, 16 Apr 2020 21:07:12 -0700

> > There's nothing wrong with using Integer to start off and then moving onto defined types.
>
> Yes there is! (see my paper at the last Ada-Europe). The first message when you teach Ada is that it is all about defining proper types. You have to start by fighting bad habits from other languages.

shrug You have your own way of looking at this. However, I really did not want to leave someone that is just starting with a long and academic chapter on types in Ada. It would be boring and discourage someone from learning Ada.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Fri, 17 Apr 2020 07:49:48 +0200

> *shrug* You have your own way of looking at this. However, I really did not want to leave someone that is just starting with a long and academic chapter on types in Ada. It would be boring and discourage someone from learning Ada.

It does not need to be long and boring. I usually tell:

"You've been taught at elementary school not to add apples and oranges. Surprisingly enough, Ada is the only language that prevents you from doing it!"

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 17 Apr 2020 14:44:27 +0200

[...]

I think the crux of the matter is who you decide your audience to be.

If you decide that you want to write a "Ada for Fortran/Pascal/C programmers" then they already know all about control structures etc; and you should dive into the type system upfront.

If, on the other hand, you are writing "Ada as a first language" (or even "Ada for Lisp programmers") then yes, by all means, postpone the discussion of user-defined types until after the basics of variables, subprograms and control structures.

So I don't think there should be real opposition; both approaches have their merits depending on the audience.

From: Jere <jhb.chat@gmail.com>
Date: Fri, 17 Apr 2020 06:07:44 -0700

> If you decide that you want to write a "Ada for Fortran/Pascal/C programmers" then they already know all about control structures etc.; and you should dive into the type system upfront.

I'll slightly disagree here. As a person who came to Ada from other languages, the first things I needed to know was how things were like for loops and if statements were handled differently in Ada. You are correct that I didn't need to learn how a for loop worked, but I did need to know how to construct a for loop in Ada. Things like "in" vs "of" for loops, how to exit loops, etc. If a person is coming from another language, I would expect the first thing to do is help them bridge the gap from where they came to where they are going: show them how to take what they already know and apply it to the new language. Then teach new features. Again, it's just my opinion, but it is based on my own experience trying to learn Ada.

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Fri, 24 Apr 2020 07:42:37 -0700

[...]

I think that there are 2 kinds of beginners to programming:

1) mathematicians at heart: beginners that start from mathematics concepts and move downward to sequential execution of mathematics (as opposed to designing soft logic circuits in FPGAs)

versus

2) electrical engineers at heart: beginners that start from hardware concepts and move upward a little to controlling that hardware then move up to abstracting that control.

There is something quite satisfying in #2 that assembly language and C typically provide to #2's adherents, hence C's popularity as “Gee whiz, mom, look at what I made the computer do” when ultimately interfacing with actual registers on an IC. Ada-for-beginners fits best here in #2 because of its focus on the need for the programmer to be aware of resource allocation (e.g., finite-sized storage pool allocation). Soon after the gee whiz phase, some people yearn fairly early on for the greater intellectual discipline and direct rich expressiveness that Ada provides instead of doing it all by wink-wink-nudge-nudge idioms in C.

There is something quite satisfying in #1 that functional languages provide to #1's adherents, hence Haskell's popularity and to a lesser extent ML in a certain older age group. The automatic memory management and arbitrarily-large bignum integers fit here, I think, because of the avoidance of thinking about the hardware very much at all.

And there might be a 3rd distinct category of beginner: those that yearn to see the world as Mealy or Moore state machines and get frustrated that neither imperative nor functional programming languages put finite state machines as the true 1st-class citizens [...]; they become attracted to Erlang and Shlaer-Mellor eventually. These people tend to reach a degree of now-this-is-what-I'm-talking-about satisfaction of sorts if they ever learn VHDL or Verilog for FPGAs, but few ever go that direction, so Erlang and Shlaer-Mellor is what they up embracing as the true maturity of their initial beginner starting point.

I have always thought that both tight Ada and loose C have been an especially uphill battle for this category of FSMphile beginner, because if FSMs are covered at all in Ada or C, it is as a passing thought in the 7th or 11th book that they read, perhaps even buried as a mere interested-reader exercise at the end of the chapter. These people are expecting FSMs to be the Hollywood star in chapter 1 of the 1st book that they read on computer programming, as a fundamental stimulus-response concept.

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Fri, 24 Apr 2020 11:35:40 -0400

>And there might be a 3rd distinct category of beginner: those that yearn to see the world as Mealy or Moore state machines [...]

In my college, state machines weren't considered something for a language class -- they were part of the (nominally) language independent /algorithms and data structures/ course.

Strangely, the only other place I've seen them recently is in the Valvano ARM Cortex-M text books (using TI TIVA-C boards).

From: Paul Rubin <no.email@nospam.invalid>
Date: Thu, 30 Apr 2020 01:01:48 -0700

> I think that there are 2 kinds of beginners to programming:
>
> 1) mathematicians at heart... 2) electrical engineers at heart

That's a very interesting way to look at things! But, I think you have to add architects, zoologists, and maybe a few other types. RMS used to say that working inside a big program was like building a city. You had to design new stuff, adapt old stuff, undertake urban renewal projects, etc. I guess that's not a beginner viewpoint though.

From: "J-P. Rosen" <rosen@adalog.fr>
Subject: Teaching Ada types
Date: Fri, 17 Apr 2020 08:17:44 +0200
Newsgroups: comp.lang.ada

(was: Re: Beginning Ada Programming, by Andrew T. Shvets (2020))

Here is how I explain the Ada approach:Whatever the language, when you need to represent data, it is important to choose an appropriate type. This involves two steps:

1) Analyze your problem to determine the requirements (range f.e.) on your type.
2) Make a representation choice, i.e. choose a machine type that covers the requirements of 1)

Ada saves you step 2), by allowing you to express your needs directly, and leaving the choice of representation to the compiler.

The trouble is that other languages offer only machine types (for basic types). Therefore, most people tend to think directly in terms of machine types and skip step 1.

Of course, this assumes that you teach how to create software, and not just a language. Some time ago, a little niece of mine started a software school. I was especially interested in how design was taught! Well, the school taught the syntax of C, and then gave programming exercises. I asked: "but how did they teach you how to go from a problem statement to its solution?" She didn't even understand what I was talking about. Sigh...

Jupyter Kernel for Ada

From: Maxim Reznik
Subject: Jupyter Kernel for Ada
Date: Mon, 6 Jul 2020 09:37:14 +0100
To: Telegram's Ada Group

[Jupyter Notebooks are interactive websites where the user can play with the code, which is given as a stream of "cells", each containing some lines. The output of each cell is interspersed, allowing impactful presentations of code and results. Users of Mathematica or Maxima will find the concept familiar. --arm]

I've made a prototype of Jupyter Kernel for Ada:

https://mybinder.org/v2/gh/reznikmm/ada-howto/master?filepath=%2Fhome%2Fjovyan%2FHello_Ada.ipynb

Anton writes:

> So, what did you use for the REPL?

Maxim writes:

The kernel uses a try and error method to combine code into something that the compiler understands. There's no separate executable underneath of the kernel, except gprbuild.

Jay writes:

> For an OS

Maxim writes:

It works on Linux, but should work on Windows/macOS also, I hope. And it works "in the clouds" using mybinder.org, so you can try it in a browser and share by a link.

Ada-related Resources

[Delta counts are from Apr 6th to Jul 20th. --arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: Mon, 20 Jul 2020 09:38:21 +0100
To: Ada User Journal readership

Ada groups on various social media:

- LinkedIn: 2_950 (+47) members	[1]
- Reddit: 4_086 (+745) members	[2]
- StackOverflow: 1864 (+69) questions	[3]
- Freenode : 88 (-7) users	[4]
- Gitter: 56 (+5) people	[5]
- Telegram: 79 (+18) users	[6]
- Twitter: 53 (-35) tweeters	[7]
65 (-104) unique tweets	[7]

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

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

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

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

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

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

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

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: Mon, 20 Jul 2020 09:38:21 +0100
To: Ada User Journal readership

GitHub: 652 (+76) developers	[1]
Rosetta Code: 747 (+40) examples	[2]
37 (-1) developers	[3]
Sourceforge: 275 (+4) projects	[4]
Open Hub: 212 (+1) projects	[5]
Bitbucket: 90 (+2) repositories	[6]
Codelabs: 51 (+2) repositories	[7]
AdaForge: 8 (=) repositories	[8]

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

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

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

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

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

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

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

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

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: Mon, 20 Jul 2020 09:38:21 +0100
To: Ada User Journal readership

[Positive ranking changes mean to go down in the ranking. The IEEE last report is kept, as the 2020 report has not yet been released. --arm]

- TIOBE Index: 43 (+6) 0.28% (+0.05%)	[1]
- IEEE Spectrum (general): 43 (=) Score: 24.8	[2]
- IEEE Spectrum (embedded): 13 (=) Score: 24.8	[2]

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

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

Searchable comp.lang.ada Archive Since 1982

From: Jeremy Grosser <jgrosser@gmail.com>
Subject: Searchable comp.lang.ada archive dating back to 1982
Date: Wed, 17 Jun 2020 19:39:37 -0700
Newsgroups: comp.lang.ada

I'm not fond of Google Groups, so I built my own archive.

https://archive.legitdata.co/comp.lang.ada/

Sources:

UTZOO tapes

1982 - 1991

https://archive.org/details/utzoo-wiseman-usenet-archive

Usenet Historical Collection

1993 - 2013

https://archive.org/details/usenethistorical

Eternal September NNTP

2012 - Current

http://www.eternal-september.org/

The earliest messages here were copied from the net.lang.ada group, which was renamed to comp.lang.ada in 1986. If you have messages from either of these groups that aren't in the archive, I'd love to include them.

Where practical, an additional Date header has been added to each message in ISO 8601 format to aid in chronological sorting. Where no timezone was given, UTC is assumed. Early messages routed via UUCP were often delayed by days as indicated by the difference between the Posted and Date-Received timestamps. In most cases, I use the value from the Posted timestamp.

A spam filter has been applied to the archive. Many thousands of advertisements for prescription drugs, sex acts, spiritual salvation, and prejudice have been removed. I do not wish to host this type of content and are actively working to train better filters and remove spam that slipped through.

This archive is updated hourly via NNTP.

From: briot.emmanuel@gmail.com
Date: Wed, 17 Jun 2020 23:41:32 -0700

> https://archive.legitdata.co/comp.lang.ada/

Impressive work! Well done

When we click on a message (for instance the very first message from Robert Dewar, https://archive.legitdata.co/comp.lang.ada/20200615230049.GPFs9s23hbJuXG9EM_R0FWXl5noNdM87UxDfojB0oVo@z/) the date doesn't seem to be visible (it is in the summary window, 1983 in this case).

I note that for most other messages it is visible though!

The first message that contains "GNAT" is from Tucker Taft in 1993. Not sure whether there was a formal announcement for the birth of GNAT.

In any case, some fun reading to be had for historically-inclined people.

From: jgrosser@gmail.com
Date: Thu, 18 Jun 2020 00:48:08 -0700

>

> When we click on a message (for instance the very first message from Robert Dewar, https://archive.legitdata.co/comp.lang.ada/20200615230049.GPFs9s23hbJuXG9EM_R0FWXl5noNdM87UxDfojB0oVo@z/) the date doesn't seem to be visible (it is in the summary window, 1983 in this case).

>

> I note that for most other messages it is visible though!

> [...]

Thanks for the feedback! I've [fixed] the missing date issue for messages where there was no Date header, only a Posted header.

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Thu, 18 Jun 2020 03:14:57 -0500

> I'm not fond of Google Groups, so I built my own archive.

>

> https://archive.legitdata.co/comp.lang.ada/

> [...]

Very nice and good job. Thanks for doing this.

Did you use Ada to do the above? Or just pure JavaScript or other software?

From: jgrosser@gmail.com
Date: Thu, 18 Jun 2020 01:34:50 -0700

>

> Did you use Ada to do the above? Or just pure javascript or other software?

At the moment, I'm using public-inbox [1], which is a horrifying mess of Perl scripts. Now that I've got all of the archives in a reasonably consistent format, I do plan to rewrite it in Ada.

The only JavaScript involved at the moment is for obfuscating email addresses from naive crawlers, which the public-inbox maintainers felt was necessary.

[1] https://public-inbox.org/README.html

Ada-related Tools

VisualAda 1.3.2

[Releases for VisualAda 1.3 and 1.3.1 were also announced in this period, with the following changes. --arm]

> - Added preliminary support for the GNAT Community edition 2019 ARM toolchain and the associated runtimes.

> - Preliminary support for "Peek Definition"

> - Improved statement completion

> - Improved symbol handling (more dynamic)

From: alby.gamper@gmail.com
Subject: ANN: VisualAda (Ada Integration for Visual Studio 2017 & 2019) release 1.3.2
Date: Sat, 27 Jun 2020 20:55:11 -0700
Newsgroups: comp.lang.ada

Dear Ada Community

VisualAda version 1.3.2 has been released

Enhancements include the following:

- Preliminary support for "Find all references"
- Fix - Static Library project template did not reference ARM targets correctly
- Add support for Tools->Options->Ada Language

Please feel free to download the free plugin from the following URL

https://marketplace.visualstudio.com/items?itemName=AlexGamper.VisualAda

Simple Components 4.49

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components v4.49
Date: Thu, 7 *May 2020 09:36:03 +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.

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

Changes to the previous version:

- Bug fix in GNAT.Sockets.Connection_State_Machine.HTTP_Client.Signaled that prevented signaling successful connection;
- Sample code Test_HTTPS_OpenSSL_JSON_Client added;
- Free_Space function added to Generic_FIFO;
- The package Generic_Bounded_Map added.

HAC 0.06

[Release 0.05 was also announced in this period, with the following changes. --arm]

> - type VString (variable-size string), with concatenation ("&" operator) including concatenation with numeric types (their image), comparison operators, Element, Length, Slice, Index, "*", Trim, Image, Integer_Value, Float_Value functions; Get_Line, Put, Put_Line subprograms for VString> - Argument_Count, Argument (the latter returns a VString)

> - Get_Env, Set_Env, Shell_Execute system subprograms

From: gautier_niouzes@hotmail.com
Subject: Ann: HAC v.0.06 - Text File I/O
Date: Tue, 19 May 2020 01:50:12 -0700
Newsgroups: comp.lang.ada

HAC (HAC Ada Compiler) is available on two open-source development sites:

https://hacadacompiler.sourceforge.io/

https://github.com/zertovitch/hac

HAC is a small, quick, open-source Ada compiler, covering a subset of the Ada language. Even though the HAC documentation is more or less non-existent, the good news is that you can use as a help Ada books and online documentation about Ada: HAC does not define a dialect of Ada, only a subset. A glimpse into the file "src/hac_pack.ads" gives you the currently available types and subprograms.

The latest additions are:

- v.0.06: Text File I/O around File_Type (example below)

HAC programs are real Ada programs, they can be built by a "serious" Ada compiler, through the HAC_Pack compatibility package. See the exm/hac_exm.gpr and test/hac_test.gpr project files for the GNAT compiler.

HAC is itself programmed in Ada. To build HAC for the command-line, all you need (with GNAT) is to run "gprbuild -p -P hac". Then you get the executable hax[.exe]. The command "hax" alone will show you basic help.

```
 HAX: command-line compilation and execution for HAC (HAC Ada Compiler)

 Compiler version: 0.06 dated 18-May-2020.

 URL: https://hacadacompiler.sourceforge.io/

 Usage: hax [options] main.adb [command-line parameters for main]

 Options: -h: this help

          -v, v1: verbose

          -v2 : very verbose

          -a : assembler output

          -d : dump compiler information
```

Enjoy

[...]

PS: The example (exm/file_copy.adb):

```
with HAC_Pack;  use HAC_Pack;

procedure File_Copy is
  s : VString;
  f1, f2 : File_Type;
begin
  Open (f1, "file_copy.adb");
  Create (f2, "file_copy.txt");
  while not End_Of_File (f1) loop
    Get_Line (f1, s);
    Put_Line (f2, s);
  end loop;
  Close (f1);
  Close (f2);
end File_Copy;
```

From: joakimds@kth.se
Date: Tue, 19 May 2020 05:39:58 -0700

> [...]

> - v.0.06: Text File I/O around File_Type (example below)

Nice Gautier regarding the possibility of copying text files. Is support for copying/reading/writing binary files in the works?

From: gautier_niouzes@hotmail.com
Date: Wed, 20 May 2020 02:47:48 -0700

> Nice Gautier regarding the possibility of copying text files.

Thx. Of course file_copy.adb is a tiny demonstration example. You can now make a parser or whatever you want with text files, from HAC: nested subprograms, local types and variables are supported.

> Is support for copying/reading/writing binary files in the works?

Not yet, but I've added this to the to-do list.

From: Stéphane Rivière <stef@genesix.fr>
Date: Wed, 20 May 2020 13:29:57 +0200

> HAC (HAC Ada Compiler) is available on two open-source development sites:

Well done Gautier!

I've build (a breeze) and test HAC: this is not a toy but a really useable tool for simple - but not simplistic - purposes...

An educative Ada way inside (the compiler code) and outside (for Ada newbies to adminsys complex scripting needs).

Straight reuse of HAC code in GNAT is really great (and a key point IMHO).

All the best from Oleron Island :)

HAC 0.07, LEA 0.71

From: gautier_niouzes@hotmail.com
Subject: Ann: HAC v.0.07, LEA 0.71
Date: Fri, 5 Jun 2020 13:34:34 -0700
Newsgroups: comp.lang.ada

In a nutshell:

- HAC (the HAC Ada Compiler) has now an exception system, with messages and trace-backs.

  Details here:

  https://gautiersblog.blogspot.com/2020/06/hac-v007-exceptions-and-trace-backs.html

- LEA (a Lightweight Editor for Ada) leverages trace-backs in its navigation system.

  Details here:

  https://gautiersblog.blogspot.com/2020/06/lea-071-with-exception-trace-back.html

HAC is pure Ada (*).

LEA is Windows only (although the LEA_Common part is pure Ada), but runs seamlessly on the Wine emulator for instance.

Enjoy!

(*) ... except for the following bit which should be easy to adapt if you build HAC with another compiler than GNAT:

```
--  Here is the non-Ada-standard stuff in
-- HAC_Pack.
package Non_Standard is
  function Sys (Arg :
      Interfaces.C.char_array) return Integer;
  pragma Import(C, Sys, "system");
  Directory_Separator : constant Character;
  pragma Import (C, Directory_Separator,
                        "__gnat_dir_separator");
end Non_Standard;
```

Gnu Emacs Ada Mode 7.1.3

[Version 7.1.1 was also announced in this period, with the following changes. --arm]

> * ada-mode fully supports non-ASCII text (the few remaining ASCII-only regular expressions have been fixed).

> * gpr_query now starts in the background, and provides a completion table of all symbols in the project.

> * keystroke C-M-i is bound to `completion-at-point', and uses the symbol table provided by gpr-query.

> * Commands that prompt for a symbol (i.e. C-u C-c C-d wisi-goto-spec/body and C-u M-. xref-find-definitions) use the completion table provided by gpr_query. With a single C-u, all symbols in the project are used; with two C-u, only symbols defined in the current file are used.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: Gnu Emacs Ada mode 7.1.3 released.
Date: Sun, 7 Jun 2020 14:22:39 -0700
Newsgroups: comp.lang.ada

Gnu Emacs Ada mode 7.1.3 is now available in GNU ELPA.

Relative to the previous Ada mode release (7.1.2), this is a bug fix release.

* There was a bug in wisi--before-change that made it miss many buffer changes.

See the NEWS files in ~/.emacs.d/elpa/ada-mode-7.1.3 and wisi-3.1.3, or at http://www.nongnu.org/ada-mode/, for more details.

The required Ada code requires a manual compile step, after the normal list-packages installation ('install.sh' is new in this release):

```
cd ~/.emacs.d/elpa/ada-mode-7.1.3
./build.sh
./install.sh
```

If you get an error like:

```
sal-gen_unbounded_definite_red_black_trees.adb:326:29: access discriminant in return aggregate would be a dangling reference
```

it is due to a bug in all recent versions of GNAT. Edit the file in ~/.emacs.d/elpa/wisi-3.1.3; see the WORKAROUND comment there. Different versions of GNAT either require the .all or forbid it.

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

GCC 10.1.0 for macOS

From: Simon Wright <simon@pushface.org>
Subject: ANN: GCC 10.1.0 for macOS
Date: Sat, 16 May 2020 13:47:09 +0100
Newsgroups: comp.lang.ada

Native and cross-arm-eabi compilers now available at https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/10.1.0/

AdaCore GNAT Community Edition 2020

From: Karl Müller <mtb23@gmx.de>
Subject: AdaCore GNAT Community Edition 2020 arrived
Date: Wed, 20 May 2020 19:27:40 +0200
Newsgroups: comp.lang.ada

To whom it may concern:

AdaCore GNAT Community Edition 2020 arrived:

https://www.adacore.com/download

From: DrPi <314@drpi.fr>
Date: Wed, 20 May 2020 23:15:20 +0200

Great!

Any known schedule for bb-runtime "community 2020"?

From: Mark Lorenzen <mark.lorenzen@gmail.com>
Date: Mon, 25 May 2020 00:00:46 -0700

> Any known schedule for bb-runtime "community 2020"?

What target do you have in mind? I can see that both the ARM and RISC-V ports of the bare-board run-time are available for download.

From: DrPi <314@drpi.fr>
Date: Mon, 25 May 2020 14:30:34 +0200

>

> What target do you have in mind? I can see that both the ARM and RISC-V ports of the bare-board run-time are available for download.

Cortex-M4 based NXP Kinetis micro-controller.

I started to write my own runtime based on Adacore bb-runtime github repository (Community 2019). It does not compile with GNAT Community edition 2020.

ISAM Implementation

From: Norman Worth <nworth@comcastNOSPAM.net>
Subject: ISAM
Date: Thu, 21 May 2020 10:53:36 -0600
Newsgroups: comp.lang.ada

Is there an Ada package or binding for ISAM [indexed sequential access method]? I seem to remember one many years ago, but I can't find anything now.

From: Wesley Pan <wesley.y.pan@gmail.com>
Date: Thu, 21 May 2020 10:55:59 -0700

I think you are referring to the old ACM SigAda Ada Letters articles by Karl Kuberl and Wolfram Pietsch on their ISAM implementation. They are on the ACM webpage.

Here are the direct links to the PDF files (need to rotate the view 90deg, btw):

"A Portable Ada Implementation of Indexed Sequential Input-Output"

[Part 1/2] https://dl.acm.org/doi/pdf/10.1145/381943.381955?download=true

[Part 2/2] https://dl.acm.org/doi/pdf/10.1145/9305.9306?download=true

I only skimmed the articles. Doesn't appear to be a way to get their implementation... Maybe email them?

From: Stéphane Rivière <stef@genesix.fr>
Date: Thu, 21 May 2020 20:09:29 +0200

> Is there an Ada package or binding for ISAM? I seem to remember one many years ago, but I can't find anything now.

Hi Norman,

From the outdated AIDE (Ada Instant Development Environment) source repository... Nancy, a very kind person - and also a remarkable modern painter - gave me permission [1], years ago, to distribute all the files from her book.

I've just re-packaged this (and more) for you and all cla ng readers:

The Nancy Packages (LGPL licence) from her book (ISAM sources - Chap 09):

https://stef.genesix.org/pub/ada/Files_structures_with_Ada.zip

Her near 500 pages book is really good (everyone interested in disk file structures and btree indexes should have this book)... for six bucks wo shipping :()

https://www.amazon.com/Structures-Benjamin-Cummings-Computer-Science/dp/0805304401

And... more here (mainly public domain but check licences - more btree, paradox db and even a well documented dbase implementation with btree index support):

https://stef.genesix.org/pub/ada/Sequential_Indexed_With_Ada.zip

Have fun.

Stef

[1] [Quoted email from Nancy Miller follows. --arm]

Stephane, Sorry to be so long in responding but we are home now and adjusting to the change in time zones. The AIDE sounds very exciting.

We would be willing to give you permission to use the source code from our book with the license of LGPL. I have completed the reference below for you.

I'm not sure the book is listed on the publishing company's web page but should be available if someone were to ask.

Les sources de ce répertoire proviennent de l'ouvrage :

The sources contained in this directory are coming from the book:

File Structures With Ada
Nancy E Miller and Charles G Petersen
Alan Apt
ISBN 0-8053-0440-1

http://www2.netdoor.com/~petersen/nembooks

http://www.aw-bc.com

Avec l'autorisation de l'auteur.

With the author's permission.

Nancy E Miller

From: Stéphane Rivière <stef@genesix.fr>
Date: Fri, 22 May 2020 09:39:01 +0200

> Here are the direct links to the PDF files (need to rotate the view 90deg, btw):

Thanks Wesley for the pointers. Rotate it and concatenate too:

https://stef.genesix.org/pub/ada/A_portable_Ada_implementation_of_index_sequential_input-output.pdf

AdaStudio 2020 Free Edition

From: leonid.dulman@gmail.com
Subject: AdaStudio 2020
Date: Tue, 26 May 2020 20:52:50 -0700
Newsgroups: comp.lang.ada

Announce: AdaStudio-2020 free edition

1. Qt5Ada is Ada-2012 port to Qt5 framework (based on Qt 5.15.0 final)

   Qt5ada version 5.15.0 open source and qt5c.dll,libqt5c.so(x64) built with Microsoft Visual Studio 2019 in Windows, gcc x86-64 in Linux (includes binaries prebuilds Qt 5.15.0).

   Package tested with GNAT gpl 2019 Ada compiler in Windows 64bit , Linux Debian 10 x86-64

   It supports GUI, SQL, Multimedia, Web, Network, Touch devices, Sensors,Bluetooth, Navigation and many others things.

   Changes for new Qt5Ada release:

   Added new package: Qt.QPDF for manipulate wit PDF documents

   The full list of released classes is in "Qt5 classes to Qt5Ada packages relation table.docx"

2. VTKAda version 9.0 is based on VTK 9.0.0 (OpenGL2) is fully compatible with Qt5Ada 5.15.0

   vtkc.dll,vtkc2.dll (libvtkc.so,libvtkc2.so) were built with Microsoft Visual Studio 2019 in Windows 10 (WIN64) and gcc in Linux Debian 10 x86-64(includes binaries prebuilds VTK 9.0.0)

3. Qt5AVAda is ada-2012 port to QtAV multimedia playback framework based on Qt + FFmpeg. Cross platform. High performance.

   Easy to use and base on QtAV 1.13 developed by wang-bin https://github.com/wang-bin/QtAV.

   QtAVAda builds widgets inside Qt5Ada application (includes binaries prebuilds QtAV 1.13 and ffmpeg 4.4.2)

4. Voice recognition package(speech2text) is a qtada extension, based on pocketsphinx .

   As a role Ada is used in embedded systems, but with QTADA (+VTKADA) you can build any desktop applications with powerful 2D/3D rendering and imaging (games, animations, emulations) GUI, Database connection, server/client, Internet browsing , Modbus control and many others thinks.

AdaStudio-2020 https://r3fowwcolhrzycn2yzlzzw-on.drv.tw/AdaStudio/adastudio.html web page

or Google drive

https://drive.google.com/folderview?id=0B2QuZLoe-yiPbmNQRl83M1dTRVE&usp=sharing (google drive. It can be mounted as virtual drive or directory or viewed with Web Browser)

I hope AdaStudio-2020 will be useful for students, engineers, scientists and enthusiasts

With AdaStudio-2020 you can build any applications and solve any problems easily and quickly.

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

Win32 and WinRT Bindings 10.0.19041

From: alby.gamper@gmail.com
Subject: Ann: Win32 and WinRt bindings update
Date: Fri, 5 Jun 2020 18:33:26 -0700
Newsgroups: comp.lang.ada

Dear Ada Community

The Win32 and WinRT bindings have both been updated to the latest Microsoft SDK version (10.0.19041). This version corresponds to the 20H1 release of Windows 10.

Packages/Source can be found at

https://github.com/Alex-Gamper/Ada-Win32

https://github.com/Alex-Gamper/Ada-WinRT

For a detailed list of what is new in the SDK refer to the following links

https://docs.microsoft.com/en-us/windows/uwp/whats-new/windows-10-build-19041

https://docs.microsoft.com/en-us/windows/uwp/whats-new/windows-10-build-19041-api-diff

PragmAda Reusable Components

From: PragmAda Software Engineering <pragmada@pragmada.x10hosting.com>
Subject: [Reminder] The PragmAda Reusable Components
Date: Sun, 7 Jun 2020 17:15:41 +0200
Newsgroups: comp.lang.ada

The PragmARCs are a library of (mostly) useful Ada reusable components provided as source code under the GMGPL 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.

Jeffrey R. Carter, President

PragmAda Software Engineering

pragmada.x10hosting.com

pragmada.tk

github.com/jrcarter

OpenGLAda 0.8.0

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Subject: Ann: OpenGLAda v.0.8.0 released
Date: Sat, 20 Jun 2020 11:28:57 +0200
Newsgroups: comp.lang.ada

Reddit user flyx86 asked that his announcement be crossposted here. The original announcement is at https://www.reddit.com/r/ada/comments/hc8de8/openglada_v080_released/

Posting: begin

This release contains a breaking change, namely the removal of the SOIL library previously used to load image files. SOIL has been removed since on macOS, it depends on the Carbon API which has been deprecated for a long time and finally removed in macOS Catalina. Being a C library with its last release in 2008, SOIL was always more of a necessary [evil] than a good part of OpenGLAda.

As replacement for SOIL, the excellent Generic Image Decoder (GID) library is included in this OpenGLAda release. Moreover, a package GL.Images has been added that uses GID to provide an API similar to the one of the removed SOIL binding.

The long deprecated FTGL binding has also been removed since a replacement has been available for some time with the FreeTypeAda binding and the simple GL.Text API built on top of that binding.

Some additional OpenGL functionality has been wrapped, see the changelog for details.

Finally, all example code has been extracted to an own repository and is not spread with OpenGLAda anymore.

I would like to thank Roger Mc Murtrie for his continued contribution of examples and core functionality.

Regarding OpenGLAda's future, it keeps being in maintenance mode and won't see new features unless someone contributes them. I am aware that quite some OpenGL 4.x functionality is missing. Asmaintainer, I will respond to issues on GitHub to keep the library usable.

end Posting;

LEA - Lightweight Editor for Ada 0.74

From: gautier_niouzes@hotmail.com
Subject: LEA - Lightweight Editor for Ada - Binary release v. 0.74
Date: Tue, 30 Jun 2020 00:42:56 -0700
Newsgroups: comp.lang.ada

LEA - Lightweight Editor for Ada - Binary release v. 0.74

LEA is a Lightweight Editor for Ada.

What's new in 2020 so far:

- embedded Ada samples collection, ready to run
- exception trace-back for HAC
- console input for HAC
- "auto-repair" feature (tool icon in the compiler message box)

Features:

- multi-document
- multiple undo's & redo's
- multi-line edit, rectangular selections
- color themes, easy to switch
- duplication of lines and selections
- syntax highlighting
- parenthesis matching
- bookmarks
- includes the HAC Ada Compiler
- free, open-source, fully programmed in Ada

Currently available on Windows.

Gtk or other implementations are possible: the LEA_Common[.*] packages are pure Ada, as well as HAC.

Web site: https://l-e-a.sourceforge.io/

Blog: https://gautiersblog.blogspot.com/search/label/LEA

Enjoy!

From: Stéphane Rivière <stef@genesix.fr>
Date: Tue, 30 Jun 2020 10:19:58 +0200

Looks like a tiny GNAT Community environment at human scale!

With a true Ada subset as HAC sources are compilable by GNAT...

Does it work with Wine?

[...]

Yes!!!!

And... All this embedded in one binary... Amazing. A real KISS Ada environment for serious scripting tasks and educational purposes.

Thanks and congrats Gautier ;)

SweetAda 0.1 Released

From: gabriele.galeotti.xyz@gmail.com
Subject: SweetAda 0.1 released
Date: Tue, 30 Jun 2020 09:34:51 -0700
Newsgroups: comp.lang.ada

Hi all.

I've just released SweetAda version 0.1.

SweetAda is a lightweight development environment to create Ada code on a wide range of CPUs and platforms.

Have a look at http://www.sweetada.org.

This is an experimental preview, so comments and advice are welcome.

Feel free to ask me about everything, I know that a lot of components are poorly documented and difficult to understand without startup information. This is a heavy work in progress, the manual is under incomplete translation and things could be not in-sync.

Many things do work, but many things, even basic, are still in a TODO list, and I will complete them upon an explicit interest.

E.G.:

- a PC-x86 platform will respond to network pings
- IBM S/390 (emulated) support is lacking even basic CPU setup, interrupts handling, etc (but will startup and write a message to a 3270 terminal)
- a Raspberry board is only able to start the first core and pulse a GPIO LED

I release SweetAda this way because otherwise I will spend other years in the development without feedback, and existing codebase shows me enough prospective.

Excuse me for slow download of the packages and toolchains, the website is hosted at my home.

Ada-related Products

PTC ObjectAda 10.2

From: Shawn M. Fanning <sfanning@ptc.com>
Subject: PTC ObjectAda V10.2 announcement for Ada User Journal
Date: Fri, Jul 24, 2020 at 11:26 PM (CET)
To: Ada User Journal readership

On July 22, 2020, PTC announced the availability of version 10.2 of our ObjectAda for Windows andObjectAda64 for Windows products. This new product release provides full support for Ada 2012 language features and represents the completion of the phased implementation strategy PTC adopted for Ada 2012 language feature support within the ObjectAda technology. With ObjectAda for Windows version 10.2, the ObjectAda compiler conforms to the Ada Conformity Assessment Test Suite (ACATS) version 4.1Q and adds several new features not present in the previous release (ObjectAda version 10.1 released in May 2019) including support for storage subpools and the Default_Storage_Pool pragma, execution time enforcement of type invariants, and complete support for new Ada expression forms.

The new installation approach introduced with ObjectAda for Windows v10.x allows ObjectAda to be used with the latest releases of Microsoft’s Visual Studio tools and the Windows 10 SDK. ObjectAda version 10.2 includes version 4.0.0 of the ObjectAda Ada Development Toolkit (ADT) Eclipse interface which supports Eclipse 2020-03 (4.15) or later. All of these upgrades combined make ObjectAda for Windows version 10.2 a solid, modern, and effective toolset for development of mission-critical application code in the Ada language. ObjectAda version 10.2 supports Ada 95, Ada 2005, and Ada 2012 compiler operation modes to provide compatibility with previous versions.

Additional information about ObjectAda version 10.2 is available within the Product Release Announcement which can be downloaded from https://www.ptc.com/products/developer-tools/objectada.

Customers with active subscription licenses for ObjectAda for Windows v10.x or ObjectAda64 for Windows v10.x are entitled to a no-charge upgrade to v10.2. (Requests for upgraded license keys can be sent by email to objectada-support@ptc.com.)

If you are not currently using ObjectAda and wish to learn more or if you are using an earlier release of ObjectAda and wish to upgrade, register your request at https://www.ptc.com/en/products/developer-tools/objectada/contact-sales.

With Best Regards,
Shawn Fanning

PTC ApexAda 5.2

From: Shawn M. Fanning <sfanning@ptc.com>
Subject: PTC ObjectAda V10.2 announcement for Ada User Journal
Date: Fri, Jul 24, 2020 at 11:26 PM (CET)
To: Ada User Journal readership

On May 19, 2020 PTC announced the release of the PTC ApexAda v5.2 Embedded for Linux/Armv8 64-bit product. This product is the initial product offering based on a new 64-bit code generator for ApexAda for the Armv8 64-bit (aarch64) architecture and is our latest release supporting 64-bit embedded application development.

The host operating system for this product is Intel x64 Red Hat Enterprise Linux v7.x/v8.x (or CentOS equivalent) distribution. Using the Linaro GNU cross-development toolchain for 64-bit Armv8 Cortex-A processors on the Linux/Intel64 host, PTC ApexAda supports the generation of Ada 95 / Ada 2005 application images that execute on ARMv8-A 64-bit (aarch64) processors (for example Arm Cortex A53, A57, A72) running 64-bit embedded Linux distributions. Examples of embedded Linux distributions which can be supported are openSUSE Leap v15.1, SUSE Linux Enterprise Server for Arm v15.1, Ubuntu Server 20.04, Wind River Linux and other Yocto-derived Linux distributions with a 64-bit kernel. Reference hardware used for the development and test of ApexAda was the Raspberry Pi 3 Model B/B+. (Raspberry Pi 4 Model B with its larger 4GB RAM configuration and other boards such as the VPX-1703 from Curtiss-Wright Defense Solutions can also be supported by ApexAda.)

Included with the 64-bit embedded compiler is the PTC® ApexAda v5.2 64-bit compiler for Linux native application development. Also included is the integrated ApexAda 64-bit C/C++ compiler which facilitates seamless development of mixed-language applications written in Ada, C, and C++. ApexAda V5.2 Embedded compilers provide a complete cross-development toolchain hosted from Linux distributions including RedHat Enterprise Edition, CentOS, and SUSE. A complete description of PTC ApexAda v5.2 Embedded for Linux/Armv8 64-bit is available within the Product Release Announcement which can be downloaded from https://www.ptc.com/products/developer-tools/apexada .

The addition of the new code generation capability for 64-bit Armv8 processors to ApexAda opens up a whole new landscape for embedded application development using ApexAda. PowerPC processors have for a long time been a design choice for our aerospace and defense customers due to their balance of performance, cost, and power characteristics. Intel processors have offered many of our customers increased performance at a cost of additional complexity and power requirements. Driven by the mobile consumer market, Arm processors provide high performance and low power advantages over Intel processors. We think these advantages combined with the flexibility provided by embedded Linux distributions and the availability of low-cost and high-performance consumer-grade development boards as well as ruggedized 64-bit Arm boards will provide substantial benefits to our customers looking to modernize existing deployed applications while mitigating risks through continued use of the same time-proven and industrial-strength ApexAda compiler technology. The 64-bit Armv8 (aarch64) processors are now well-known and proven processors with a long lifecycle and there are multiple 64-bit Linux distributions available which run on these processors. Follow-on products leveraging the new ApexAda 64-bit Armv8 (aarch64) code generation capability for other real-time operating systems are under development with prioritization based on customer interest and requirements.

If you would like to receive additional information about the new PTC ApexAda v5.2 Embedded for Linux/Intel64 to Linux/Armv8 64-bit product or wish to be contacted by a PTC Developer Tools sales representative regarding evaluations, upgrades and associated pricing, register your request at https://www.ptc.com/en/products/developer-tools/objectada/contact-sales .

With Best Regards,
Shawn Fanning

Ada and Operating Systems

Scheduling Behaviour Issue with FreeRTOS

From: Simon Wright <simon@pushface.org>
Subject: Scheduling behaviour issue
Date: Wed, 22 Apr 2020 12:34:48 +0100
Newsgroups: comp.lang.ada

As some will recall, I've based my Cortex GNAT RTS[1] (for ARM Cortex-M devices, so far) on FreeRTOS[2].

I've now discovered an unfortunate difference between what the ARM requires at D.2.3(9)[3] and the way FreeRTOS behaves. What we need is

"A task dispatching point occurs for the currently running task of a processor whenever there is a nonempty ready queue for that processor with a higher priority than the priority of the running task. The currently running task is said to be preempted and it is added at the head of the ready queue for its active priority."

but FreeRTOS adds the preempted task at the *tail* of its ready queue ([4], section Prioritized Pre-emptive Scheduling (without Time Slicing), on page 95 or thereabouts).

I can see that this will make an application less predictable, but I don't think it'll make a correct application misbehave.

I've been having some trouble thinking of a way to demonstrate the (mis)behaviour!

[1] https://github.com/simonjwright/cortex-gnat-rts

[2] https://www.freertos.org

[3] http://www.ada-auth.org/standards/rm12_w_tc1/html/RM-D-2-3.html#p9

[4] https://bit.ly/2VK7slM

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Wed, 22 Apr 2020 21:03:53 +0300

> [Original message.]

I'm not sure about the definition of a "correct [Ada] application" in this context, but it seems to me that the Ada RM rule means that if several tasks have the same priority, they can assume mutual non-pre-emption, in essence that the running task will not yield to another task within this same-priority set until the running task explicitly blocks or yields.

Under that rule, therefore, tasks at the same priority, on the same processor core, can act on shared data without mutual-exclusion protections -- more or less as in a co-operative, non-pre-emptive system -- even if they are pre-empted by higher-priority tasks (which do not share these same data). The tasks in the same-priority set just have to take care not to block or yield while engaged in such actions on shared data.

Under RTEMS, if there are higher-priority tasks on that processor core, such actions on shared data would not have this mutual-exclusion property, and the shared data could be messed up. However, I'm not sure if such use of shared data is "correct" per the Ada RM, and if the resulting mess can be called "misbehaviour".

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Wed, 22 Apr 2020 13:41:19 -0700

> Under RTEMS, if there are higher-priority tasks on that processor core, such actions on shared data would not have this mutual-exclusion property, and the shared data could be messed up. However, I'm not sure if such use of shared data is "correct" per the Ada RM, and if the resulting mess can be called "misbehaviour".

This is the reasoning of Ada 83 (RM 9.11 Shared Variables):

<quote>

For the actions performed by a program that uses shared variables, the following assumptions can always be made:

* If between two synchronization points of a task, this task reads a shared variable whose type is a scalar or access type, then the variable is not updated by any other task at any time between these two points.

* If between two synchronization points of a task, this task updates a shared variable whose type is a scalar or access type, then the variable is neither read nor updated by any other task at any time between these two points.

The execution of the program is erroneous if any of these assumptions is violated.

</quote>

I'm too lazy to search for the relevant text in current Ada, but as far as I can tell, this principle is still valid. This is one of the reasons I guess that protected objects were introduced.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 23 Apr 2020 00:58:49 +0300

> This is the reasoning of Ada 83 (RM 9.11 Shared Variables):
>
> [snipped]
>
> I'm too lazy to search for the relevant text in current Ada, but as far as I can tell, this principle is still valid.

The wording in Ada 2012 is very different (RM 9.10 Shared Variables, and C.6 Shared Variable Control).

As I understand it, two tasks can read and/or write shared atomic variables without that being erroneous, and all tasks see the same order of operations on each variable separately, but the interleaving order of these reads and writes from the two tasks is not specified (if the reads/writes are not in a protected operation or similar).

Mutual exclusion for actions on shared data is usually necessary to ensure that a _sequence_ of reads/writes done by one task is not _interleaved_ with reads/writes from another task. As far as I can see, RM 9.10 and C.6 (and Ada 83 RM 9.11) do not address that question.

The usual example is a simple increment of an atomic counter variable (read - add one - write):

    Counter := Counter + 1;

which can lose one count if executed interleaved by two tasks. However, if the two tasks have the same priority, and compete for the same processor, and the Ada rule quoted by Simon (D.2.3(9)) applies, then both tasks can execute the increment without risking interleaving, or so it seems to me. But if the RTEMS rule is followed, then pre-emptions from higher-priority tasks can force interleaving of instructions from the two incrementing tasks, and thus break the counter.

> This is one of the reasons I guess that protected objects were introduced.

Certainly (and why rendez-vous parameters were introduced in Ada 83) and I would definitely recommend using protected objects for shared counters. That would make the counters work properly even under RTEMS.

Simon's challenge was to find a correct program that misbehaves under the RTEMS scheduling rule. I think my example will misbehave (not work as the programmer expected) but I'm not fully sure if the Ada RM defines its behaviour even under the Ada rules. I'm looking for an RM rule that says that if two tasks have the same priority and are scheduled on the same processor then only one task is running at a time, and executes all its reads/writes without any interleaved reads/writes from the other task, until the running task somehow yields to the other task. This may be implicit in the notions of "scheduling" and "running", but I would prefer an explicit connection between those notions and RM 9.10 and C.6.

In this connection I want to ask if the "Discussion" in RM 2012 9.10(15.b) uses a valid example. The Discussion says that two "sequential" assignments to the same variable, where neither "signals" the other, are not erroneous, because there may be cases where the order in which the assignments are executed makes no difference. The Discussion gives, as an example, assignments that just "accumulate aggregate counts". It seems to me that the order of two such assignments to the same counter does matter, because the values written may be different, as in the counter-increment example above. Am I right? If so, this example seems wrong for this Discussion (also in Ada 202X ARM).

From: Simon Wright <simon@pushface.org>
Date: Thu, 23 Apr 2020 12:48:21 +0100

I've done some tests on this with GNAT CE 2019. Test program at the end; the commented-out lines are for checks on host machines, irrelevant for single-cpu STM32F4 boards. The test execution is to be under GDB, but a breakpoint on the null; in task C (line 48), and set up this script for that breakpoint:

```
command
silent
print As
print Bs
continue
end
```

On macOS with 4 CPUs, both As and Bs are updated, and the user load is ~199% (i.e. two CPUs in use).

On debian stretch under VMware (1 CPU), both As and Bs are updated.

Conclusion: the macOS host RTS doesn't respect the CPU restriction. Can't tell about macOS, but the Linux RTS behaves in the same way as FreeRTOS.

On STM32F4, with cortex-gnat-rts, the behaviour is as I expected (both As and Bs updated).

On STM32F4, with ravenscar-{sfp,full}-stm32f4, the behaviour is as D.2.3(9)[3] (only As updated).

```
with Ada.Real_Time;
with System;
--  with System.Multiprocessors;

package body Priority_Issue is

   type Count is mod 2 ** 64;
   As : Count := 0;
   Bs : Count := 0;

   task A
   with
     --  CPU => 1,
     Priority => System.Default_Priority;

   task B
   with
     --  CPU => 1,
     Priority => System.Default_Priority;

   task C
   with
     --  CPU => 1,
     Priority => System.Default_Priority + 1;

   use type Ada.Real_Time.Time;

   task body A is
   begin
      delay until Ada.Real_Time.Clock +
            Ada.Real_Time.Milliseconds (300);
      loop
         As := As + 1;
      end loop;
   end A;

   task body B is
   begin
      delay until Ada.Real_Time.Clock +
            Ada.Real_Time.Milliseconds (600);
      loop
         Bs := Bs + 1;
      end loop;
   end B;

   task body C is
   begin
      loop
         delay until Ada.Real_Time.Clock +
               Ada.Real_Time.Seconds (1);
         null;  --  break here, print As, Bs
      end loop;
   end C;
end Priority_Issue;
```

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Thu, 23 Apr 2020 15:57:54 +0300

[...]

As I said in a later post, I agree that using static priority assignments to ensure synchronization or mutual exclusion between tasks is fragile, and more robust methods (protected objects) are preferable.

On the other hand, I often see posts on e.g. comp.arch.embedded from people who prefer co-operative multi-tasking (or even single-thread programming). They could find it attractive to use such designs based on this Ada feature.

Several coding standards (rulebooks) or design standards that I've seen have rules forbidding tasks of equal priority, I assume in order to avoid uncertainties about execution order, including the case under discussion.

Status of GNAT on Red Hat/Fedora

From: reinert <reinkor@gmail.com>
Subject: Ada (gnat) on Red Hat Enterprise og Fedora is OK ?
Date: Wed, 3 Jun 2020 08:20:14 -0700
Newsgroups: comp.lang.ada

Hello,

Has RedHat been more Ada (GNAT) friendly the latest years? Some years ago I went from Fedora to Debian because I did not manage to use "gprbuild projectfile.gpr" etc under Redhat.

I ask because someone would like to invest in using my Ada-program under RedHat or Fedora which the IT-department there prefers. Hence it would be nice if someone here could share their experience with Ada and RedHat.

From: "Dmitry A. Kazakov"
<mailbox@dmitry-kazakov.de>
Date: Wed, 3 Jun 2020 18:33:35 +0200

I cannot tell for RedHat but Fedora and CentOS are perfectly OK. Fedora has the latest GCC 10.

BTW on Raspberry for my projects Fedora beats Debian in compile/build time. Before GCC 10 it was the reverse.

From: reinert <reinkor@gmail.com>
Date: Wed, 3 Jun 2020 11:31:32 -0700

>

> I cannot tell for RedHat but Fedora and CentOS are perfectly OK. Fedora has the latest GCC 10.

And is gprbuild included in a Fedora main repository (and functions)?

From: "Dmitry A. Kazakov"
<mailbox@dmitry-kazakov.de>
Date: Wed, 3 Jun 2020 21:32:12 +0200

> And is gprbuild included in a Fedora main repository (and functions)?

Yes it does.

Under CentOS you might be required to compile gprbuild and create a configuration file for it because it does not find the compiler.

But, as I said, Fedora works out of the box.

Ada and Other Languages

Running Python Scripts from Ada

From: "Rego, P." <pvrego@gmail.com>
Subject: Running a simple Python from Ada program
Date: Fri, 3 Apr 2020 09:57:14 -0700
Newsgroups: comp.lang.ada

Does someone have a simplest as possible way to run a Python script from the Ada project?

The Python script in this case only has a print in a loop, with a sleep, so each iteration it should print the arguments.

The Python script I'm using is

```
import sys
import time
for index in range(10):
   print(sys.argv)
   time.sleep(1)
```

One approach I am trying is running this one as a batch script, so using

```
import sys
import time

with Text_IO;
with Interfaces.C; use Interfaces.C;
procedure systest2 is
   function Sys (Arg : Char_Array)
      return Integer;
   pragma Import(C, Sys, "system");
   Ret_Val : Integer;
begin
   Ret_Val := Sys(To_C
               ("python testpy.py arg1 arg2"));
end systest2;
```

The problem is that the execution blocks the script, meaning that the Python printouts are only printed at the end of the execution, at once.

I know that there is a solution (to run Python from Ada) based on GNATCOLL, but I couldn't find any example to run it.

From: "Dmitry A. Kazakov"
<mailbox@dmitry-kazakov.de>
Date: Fri, 3 Apr 2020 19:35:23 +0200

> Does someone have a simplest as possible way to run a Python script from the Ada project?

There is no simple way of doing that. Python has serious issues. [From another reply by the sender: “Loading the Python library and initializing the environment is challenging under Windows. Tasking is a problem, cleaning up is a problem, some of C API functions do not work.” --arm]

> The Python script in this case only has a print in a loop, with a sleep, so each iteration it should print the arguments.

Yes, this is what I am doing. I also keep state between iterations so that the called Python script could update an object and then get it back on the next iteration.

> The Python script I'm using is [...]. The problem is that the execution blocks the script, meaning that the Python printouts are only printed at the end of the execution, at once.

Do not do that. There exist Python C API, which you should use:

https://docs.python.org/3/c-api/index.html

> I know that there is a solution (to run Python from Ada) based on GNATCOLL, but I couldn't find any example to run it.

Well, as an alternative, you could take a look how “MAX! Home Automation” runs Python scripts.

http://www.dmitry-kazakov.de/ada/max_home_automation.htm#5.1

It:

1. Loads Python DLL dynamically
2. Initializes Python environment
3. Compiles script file and loads it as a module into the Python environment
4. Calls a function from the module (in a loop)
5. Handles exceptions from Python
6. Makes some Ada subroutines callable from the Python script

I must warn you, it is complicated. The files py.ads/adb are Python bindings. Subdirectories Linux and Windows contain OS-dependent bodies of Python library loader py-load_python_library.adb. py-elv_max_cube.ads/adb is a module of Ada subroutines callable from Python.

From: "Dmitry A. Kazakov"
<mailbox@dmitry-kazakov.de>
Date: Fri, 3 Apr 2020 22:06:49 +0200

> But I am doing this [...] which blocks the testpy.py script until its end. This should not happen.

It must, actually:

https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/system-wsystem?view=vs-2019

> So, please, how could I fix this?

You need to spawn Python in an asynchronous process or else a batch script with something like "call python test.py ddddd" inside.

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Fri, 03 Apr 2020 21:48:36 -0400

>But I am doing this [...] which blocks the testpy.py script until its end. This should not happen. So, please, how could I fix this?

system() block until the invoked command completes... That happens for both C and via the Ada definition.

However, output via python print() might be getting buffered if the spawned command does not see stdout as a console -- which might be a result of not having the C I/O environment initialized...

Try adding the -u option to the invocation https://docs.python.org/3/using/cmdline.html#miscellaneous-options

"""

-u

Force the stdout and stderr streams to be unbuffered. This option has no effect on the stdin stream.

See also PYTHONUNBUFFERED.

Changed in version 3.7: The text layer of the stdout and stderr streams

now is unbuffered.

"""

EG: "python -u testpy.py arg1 arg2"

Autogeneration of Ada Bindings from Complex C Headers

From: hreba <f_hreba@yahoo.com.br>
Subject: -fdump-ada-spec: "FILE" not declared
Date: Tue, 7 Apr 2020 19:10:45 +0200
Newsgroups: comp.lang.ada

I am trying to generate Ada bindings for the GSL (Gnu Scientific Library) odeiv2 package (ordinary differential equations). So I do the following 2 steps:

1. Go to an empty directory "src" and execute

```
g++ -c -fdump-ada-spec -C
/usr/include/gsl/gsl_odeiv2.h
```

2. Go to an empty directory "obj" and execute

```
gcc -c -gnat05 ../src/*.ads
```

Unfortunately, gsl_odeiv2.h includes stdio.h, and this leads to a series of errors like
stdio_h.ads:117:69: "FILE" not declared in "x86_64_linux_gnu_bits_types_FILE_h"

[...]

Any idea?

From: Per Sandberg <per.s.sandberg@bahnhof.se>
Date: Tue, 7 Apr 2020 21:29:00 +0200

I recalled that I played with it some years ago so I just pushed my play to github.

Have a look on:
https://github.com/Ada-bindings-project/ada-gsl

From: "Luke A. Guest" <laguest@archeia.com>
Date: Wed, 15 Apr 2020 16:18:27 +0100

[...]

It's best to use the slim variant [-fdump-ada-spec-slim] of that option, then go in and hand massage the generated code to be "nice" and not the mess you get.

Julia for Next-Generation Airborne Collision Avoidance System

From: Jerry <list_email@icloud.com>
Subject: Julia for Next-Generation Airborne Collision Avoidance System
Date: Tue, 7 Apr 2020 21:45:05 -0700
Newsgroups: comp.lang.ada

You can find these words

Safer Skies

The Federal Aviation Administration is using Julia to develop the Next-Generation Airborne Collision Avoidance System at this link

https://juliacomputing.com/case-studies/lincoln-labs.html.

Do they plan to field a final product in Julia? The article is unclear on this point.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Wed, 8 Apr 2020 08:57:08 +0200

Jets do not fly anymore. So why not? (:-))

P.S. I made Ada bindings to Julia, for I wished to try it as an alternative to Python for scripting.

Unfortunately I could not use Julia bindings beyond rudimentary tests, because Julia builds are incompatible with MinGW. If you have some third-party libraries in C they will collide with Julia's C run-time.

And Julia cannot be built from sources under MSYS either. Apparently it could be some years ago, but then they broke something and it does not work anymore. It's serious avionics stuff... (:-))

Furthermore Julia does not have loadable modules one could recompile/load once and call multiple times without compiling them each time. I would not try to use it in a medium to large size system regardless of the language qualities, which are not brilliant either.

All in one, an obvious candidate for safer skies...

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Wed, 8 Apr 2020 16:51:47 -0500

> All in one, an obvious candidate for safer skies...

There is a trend going on in software engineering for the last 30 years.

Languages that are weak on typing (no typing at all, Duck typing, loose typing, dynamic typing, etc...) are getting very popular and languages that have strong static type checking which allows more error to be detected at compile time, are being ignored and are less popular with the masses.

Go figure.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Thu, 9 Apr 2020 07:44:53 +0200

> There is a trend going on in software engineering for the last 30 years. [...]

I'd even say that the trend is for ease of writing rather than ease of reading/maintaining. Well, software engineering is not the only domain where advertising for long term benefit against immediate gain is difficult and unpopular...

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Thu, 9 Apr 2020 08:23:54 -0700

> I'd even say that the trend is for ease of writing rather than ease of reading/maintaining. Well, software engineering

Would you really call this SE?

> is not the only domain where advertising for long term benefit against immediate gain is difficult and unpopular...

“When Roman engineers built a bridge, they had to stand under it while the first legion marched across. If programmers today worked under similar ground rules,they might well find themselves getting much more interested in Ada!”

Robert Dewar

Ada Practice

Learning from Intermediate Representation

From: foo wong <crap@spellingbeewinnars.org>
Subject: Intermediate Representation
Date: Wed, 1 Apr 2020 04:40:22 -0700
Newsgroups: comp.lang.ada

Hi everyone, my real name is not Foo, it is Patrick, just keeping Google off my trail.

I have been using GnuCOBOL extensively since 2013. One thing that I love about it is that it compiles to intermediate C.

If you write a program, you can compile it to this, run ctags on the runtime and the intermediate C and then hop around jumping from the C function calls generated into the runtime to see how they are actually implemented.

I would like to do the same with Ada. Is there a way? Using readelf, I was able to get some clues and looking at the .ali files I had a few more clues but so far it does not seem to be the same thing.

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Wed, 1 Apr 2020 05:06:55 -0700

https://www.cse.iitb.ac.in/~uday/courses/cs324-05/gccProjects/node4.html
gives some command-line options that you might find interesting. Note that that webpage has a typo: it misspells GIMPLE as SIMPLE in one place, but then goes on to spell it correctly in the command-line flag name. GIMPLE is the AST primarily purposed for C/C++ to which Ada gets tree-transducer in GNAT.

From: Simon Wright <simon@pushface.org>
Date: Wed, 01 Apr 2020 16:53:08 +0100

> [Original post]

Try compiling with -gnatG; I found this very helpful when developing Cortex GNAT RTS. You have to keep your wits about you!

Look at [1], from section Tasking. This was with GCC 4.9.1 for a restricted runtime: the details, particularly task creation, vary a bit between compiler releases.

[1] https://forward-in-code.blogspot.com/2015/06/building-runtime-system-for-arm-eabi.html

From: Per Sandberg <per.s.sandberg@bahnhof.se>
Date: Wed, 1 Apr 2020 18:27:43 +0200

As others mentioned GNAT/GCC is open source so just download and read. And to get the "expanded" Ada code just

$gcc -c -gnatD source.adb

and you will get an source.adb.dg that will contain an intermediate code that is feed further into the compiler.

From: Anh Vo <anhvofrcaus@gmail.com>
Date: Wed, 1 Apr 2020 10:17:52 -0700

If using GPS, from editing pane right click -> Expanded Code -> Show entire file for example. Finer option can be chosen as well, also.

From: Bob Duff <bobduff@example.com>
Date: Wed, 01 Apr 2020 18:28:38 -0400

> Whoo! I like -gnatD and -gnatG

I like -gnatDGL. The "L" intersperses the Ada source code with the generated code.

Proposal: Auto-allocation of Indefinite Objects

From: Stephen Davies <joviangm@gmail.com>
Subject: Proposal: Auto-allocation of Indefinite Objects
Date: Fri, 3 Apr 2020 15:48:41 -0700
Newsgroups: comp.lang.ada

Firstly, apologies if this has already been discussed or, more likely, if it's a really stupid idea for some reason that I haven't thought of.

My proposal is that it should (sometimes?) be possible to declare objects of indefinite types such as String and have the compiler automatically declare the space for them without the programmer having to resort to access types.

Benefits:

1. Easier, especially for newbies/students.
2. Safer due to reduced use of access types.
3. Remove the need to have definite and indefinite versions of generic units.

It is the 3rd reason that initially got me thinking about this. It seems excessive to have two versions of packages just because one version can say "Node.Item:= New_Item;" but the other has to say "Node.Item_Ptr := new Element_Type'(New_Item);".

It's probably not a good idea for auto-allocation to be the default behaviour, so I suggest something like:

```
type Node_Type is record
   Item : new Element_Type;
   Prev : Node_Ptr_Type;
   Next : Node_Ptr_Type;
end record;
```

If Element_Type is a definite type in the instantiation then Node.Item will be a normal object of that type. Otherwise, it is implemented as a pointer but the code still treats it as an object. The target of the pointer is allocated on assignment of the object. The pointer cannot be copied to any other object. Assignments of the whole record will perform a deep-copy of the auto-allocated component. The target of the Node.Item pointer can be auto-deallocated when Node goes out of scope or is deallocated.

Ok, I've probably missed something obvious and have been wasting my time, but at least I've got plenty of time to waste at the moment.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Sat, 4 Apr 2020 10:31:35 +0200

> [Explanation of the proposed syntax omitted. --arm]

This looks interesting to me. There is a huge number of cases I am using this schema, especially when Item is initialized once.

The major advantage is of course in having a plain String instead of Unbounded_String. No conversions, no space/time penalties. I refrain from using Unbounded_String as much as possible.

Also there must be a possibility to specify the pool of Item. I frequently place things like Node_Type into an arena pool, so I want the string going there as well. Another case is marshaling such objects, so that the body of Item would not be left behind.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Sat, 4 Apr 2020 12:54:44 +0200

[...]

For an issue related to the OP's idea, consider

```
with System;
procedure Boom is
   type Very_Large_Item is ...;
   type Very_Large_Index is mod
System.Max_Binary_Modulus;
   type Very_Large_List is array
(Very_Large_Index range <>) of
Very_Large_Item;

   Last : constant := ...;

   List : Very_Large_List (0 .. Last);
begin -- Boom
   ...   -- Do some thing useful with List
end Boom;
```

There exists a value N > 0 such that Last = N works and Last = N + 1 results in Storage_Error. The actual value of N may vary depending on the compiler, target, and the actual machine on which the program is executed.

If you want to handle a List with Last > N, you have to make it an access to Very_Large_List unless you care where it is allocated. There is still a value M which will result in Storage_Error, but on most machines where you'd try to process such a large object, M >> N because on such machines the heap is much larger than the stack. Implicit dereferencing makes this change less painful than it would be without implicit dereferencing, but there are still usually places where explicit dereferencing will be needed, so there is still some pain involved even though you don't care where the object is allocated.

It would be nice if there were a compiler option where objects that don't fit on the stack would be automatically allocated on the heap, and automatically deallocated when they go out of scope.

Similar arguments can be made for a compiler option where all numeric types would be accepted, with some implemented in terms of the compiler's ability to calculate static expressions exactly, rather than the user having to switch from a numeric type to an unbounded-number pkg. This has the added value that such pkgs usually lose the automated checks that numeric types have.

All of these issues have been around for some time, and the ARG is aware of them and has chosen to take no action. That seems unlikely to change.

From: Stephen Davies <joviangm@gmail.com>
Date: Sat, 4 Apr 2020 13:55:16 -0700

[Brackets in the following quotes by the author. --arm]

>>

>> Item : New String; [ill-thought-out proposal]

On 2020-04-04, Stephen Leake wrote:

>

> [Ada-101 stuff ;-)]

On 2020-04-04, Dmitry A. Kazakov wrote:

>

> This looks interesting to me. There is a huge number of cases I am using this schema, especially when Item is initialized once.

Woohoo, I'm not a complete idiot.

On 2020-04-04, Jeffrey R. Carter wrote:

>

> the ARG is aware of them and has chosen to take no action. That seems unlikely to change.

Oh. :-(

Printing Access Values

From: ldries46 <bertus.dries@planet.nl>
Subject: Put the access value
Date: Tue, 14 Apr 2020 09:15:49 +0200
Newsgroups: comp.lang.ada

I have a situation in which it is not practical to use the debugging possibilities of GNAT GPS. Instead I want to use the Put procedure to show certain values.

Normally that is not a problem with the Predefined Language attributes (mostly with the 'image attribute).

Now I have the following:

[Redacted code for conciseness. --arm]

```
type Buffer_Pointer is access
  Block_Buffer;
```

I just want to see if the routing of the different Buffer_Pointer's is correct so I thought Buffer_Pointer'Image would show the value of the pointer, f.i. ?x000000 for null or even the simple decimal value 0.

This construction creates a failure during compiling. Should I use another attribute or some other construction?

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Tue, 14 Apr 2020 09:42:53 +0200

In Ada, a pointer is not an integer and has no 'Image attribute. A pointer is not an address either. Of course, for debugging you can indulge yourself to constructs that would be thrown at for long term maintenance. So...

1) Use Unchecked_Conversion to convert it to an appropriate integer type

2) use package Address_To_Access conversion to convert your pointer to an address, then System.Storage_Elements.To_Integer to convert the address to Integer_Address, which is an integer type.

From: briot.emmanuel@gmail.com
Date: Wed, 15 Apr 2020 00:20:31 -0700

The approach I tend to use is using `System.Address_Image`:

```
  El: Buffer_Pointer := LastBuffer;
  ...
  if El /= null then
     Ada.Text_IO.Put_Line
        (System.Address_Image
                      (El.all'Address));
  end if;
```

or if this is for slightly longer term

```
  function Convert is new
Ada.Unchecked_Conversion
    (Buffer_Pointer, System.Address);
  Ada.Text_IO.Put_Line
(System.Address_Image (Convert (El));
```

This is really just for quick debugging, and the code is never (really, I swear) committed... Otherwise, I would go to the trouble of creating an `Image` function and use that

From: Robert A Duff <bobduff@TheWorld.com>
Date: Mon, 20 Apr 2020 19:02:26 -0400

> In Ada, a pointer is not an integer and has no 'Image attribute.

Sure it does. ;-)

So do records and everything else. See AI12-0020-1 (don't pay attention to details; they're changing). I implemented that recently, so the latest development version of GNAT has it.

This program:

```
with Ada.Strings.Text_Output.Formatting;
use Ada.Strings.Text_Output;
procedure Access_Image is
  type R is record
    This : Integer;
    That : String (1..10);
  end record;
  type A is access all R;
  X : A := new R'(This => 123, That =>
"helloworld");
begin
  Formatting.Put ("\1, \2\n", X'Image,
X.all'Image);
end Access_Image;
```

```
prints:
(access 162b740),
(this =>  123,
 that => "helloworld")
```

Script-like Jobs in Ada (Ideas for HAC)

From: gautier_niouzes@hotmail.com
Subject: Script-like jobs in Ada (ideas for HAC)
Date: Fri, 24 Apr 2020 12:45:33 -0700
Newsgroups: comp.lang.ada

It's a poll - sort of.

The question is: what kind of jobs do you prefer to let a scripting language do and not Ada?

My guess is that this kind of choice is related to toolsets, libraries and other aspects that are not necessarily related to languages.

For instance you would perhaps avoid using Ada for a 30-lines program that browses files and collects some information in those files (just an example). Or a little math / stats program? Or a small game?

But maybe the real reason (even if it is unconscious) you'd avoid Ada is because the compiler you are using is slow, or because it spits objects and executable files each time you change your small program. And you'd be more comfortable with a script that runs immediately without making garbage files.

A goal of the HAC compiler [1] is precisely to blur the border between a script and an Ada program: from the command line, you write "hax myprog.adb" and it just runs (that's already working). Now I'd be curious about examples of scripts you'd consider writing in Ada with HAC, if it supplied the convenient functions and libraries. It's all work-in-progress currently, and ideas of applications are welcome at this point of the development.

Thanks for any input!

[1] http://hacadacompiler.sf.net/ and https://github.com/zertovitch/hac

From: cantanima.perry@gmail.com
Date: Fri, 24 Apr 2020 16:22:34 -0700

What about REPL with hot code reloading?

I know this is the province of LISP and recently Nim did it too, and I guess most scripting languages do it, like iPython? Anyway, I mean this: long ago I used a programming language called Basic09 that despite its name had a lot in common with Pascal and Modula-2 -- well, OK, with Ada if you want to put it that way: it offered structured programming techniques and modular programming, including modules. Like any BASIC, you interacted with an interpreter, but with Basic09 you could compile to I-code, save, load, and I believe even reload that I-code, etc.

I've always wished that modern compiled languages allowed one to do something like that, so that you could combine the best of a compiled language with the best of an interpreted one. The fact that it's pretty rare probably shows how little I know, though.

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Fri, 24 Apr 2020 19:11:41 -0500

> [Entire original post removed. --arm]

Main advantage of scripts, and by this I really mean bash, sh, Python, and maybe Perl, is that these come pre-installed on Linux. I.e. once you install Linux, most likely you'll have bash, sh, Python interpreter ready to be used. (For Windows, it will be DOS scripts)

Hence any script written will run as is, with no need to have to first "compile" it for that specific version of the OS. Also easy to email one a script and they run on their end. No need to install a compiler first, figure how to compile it, link it, etc...

That is I think is the main advantage of scripts over compiled languages. Ease of use.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sat, 25 Apr 2020 11:52:08 -0700

[...]

Reliability/longevity is the biggest problem with "scripting" languages, and with many "modern" platforms. That Visual Basic game broke with each release of Visual Basic, until I got tired of maintaining it. I have a small music playing app I wrote for Android; it breaks with each release of Android. I never did get the car dashboard controls working after they broke the first time.

From: mockturtle <framefritti@gmail.com>
Date: Sat, 25 Apr 2020 23:49:09 -0700

> The question is: what kind of jobs do you prefer let a scripting language do and not Ada?

The threshold is fuzzy, but usually I go with a script if

* I need to interact heavily with the system (e.g., moving files, directories, ...)
* I can do everything with basic shell commands. For example, if I need to do some math like matrix algebra I do not use a script (not entirely true... Once I did this by calling octave from the script and piping into it the required computations... I would not advise it, though...)
* It is something quite simple and/or I need it "fast and dirty" (e.g., to do something to lots of files, but a "something" that I will not need in the future) so that the maintenance advantage that Ada gives you has not much impact.

Sometimes in the gray area of problems too complex for a bash script, but not enough to justify the investment of initial effort required by Ada, I go with Ruby that is, IMHO, a fair compromise.

From: *Simon* Wright <simon@pushface.org>
Date: Sun, 26 Apr 2020 15:49:09 +0100

> The question is: what kind of jobs do you prefer to let a scripting language do and not Ada?

The problem was a membership database, which as inherited was an Excel spreadsheet with one sheet per year, and many issues of consistency and form.

My first thought was Ada, but I totally failed at the "simple" task of reading/writing a CSV file; looking at SQLite interfacing, producing a GUI, sending mail, and the thought that someone else might well inherit it from me, I decided that Python was the best compromise.

~3000 lines (including blanks).

From: Bojan Bozovic <bozovic.bojan@gmail.com>
Date: Mon, 27 Apr 2020 11:50:46 -0700

> The question is: what kind of jobs do you prefer to let a scripting language do and not Ada?

Make it embeddable! That's the advantage of scripting language, code/modules in compiled language can do all the heavy lifting, while providing ease of use of a scripting language.

That's normal for the web and it's the way PHP, Python and Perl work. That's also normal in game development, where game logic is often written in some scripting language.

Also for some uses it would be useful if it could run compiled pseudocode, in a manner NET does.

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Mon, 27 Apr 2020 12:01:44 -0700

> Make it embeddable! [...]

HAC's MIT license is conducive to embeddability. Conversely, GNAT would be embeddable (ignoring all technical impracticalities thereof) only in GPLv3-licensed derivative works. So that would be a difference that makes a difference in mission for HAC (no matter where it ends up 20 years from now feature-wise/language-coverage-wise).

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 27 Apr 2020 22:31:38 +0200

> Make it embeddable!

Yes, that is the key feature. On top of that:

1. Asynchronous aborting of the running script with data cleanup.
2. External loadable module/packages for it written in Ada.
3. Means to maintain the process state between calls to the script. For Python it is resolved by returning an object from the script. The object is then passed as an argument by the next call. For an Ada script one could do it better, as a kind of "library package".

BTW, there is another project alike, AdaScript: http://www.pegasoft.ca/docs/sparforte12/doc/ref_adascript.html

which also lacks the above. It is a shame that GPS uses Python for the purpose. I am using Python too, because presently nothing is better. I considered Lua and Julia, but neither were usable.

Having yet another shell is not interesting. From my experience no regular task deserves writing it in a script. Each time I do this in bash etc, I get punished for it. When I am lucky I rewrite that in Ada. I am too lazy to do this from the start always hoping it would end differently. When I am not so lucky I am stuck for years with maintaining the crap which periodically stops working.

From: Jerry <list_email@icloud.com>
Date: Tue, 28 Apr 2020 01:51:50 -0700

HAC in Jupyter(Lab)? https://jupyter.org/

From: joakimds@kth.se
Date: Wed, 29 Apr 2020 08:47:41 -0700

> The question is: what kind of jobs do you prefer to let a scripting language do and not Ada?

Hi Gautier,

I currently use Ada for scripting and need to compile "the script" before being able to execute it. I would find being able to use HAC for scripting very useful. The only issue I can see with it is the "with HAC_Pack; use HAC_Pack;" that seems to be required to use HAC. I would like to keep the "Ada scripts" cross-compiler if possible.

From: gautier_niouzes@hotmail.com
Date: Thu, 30 Apr 2020 01:02:43 -0700

Thank you all for your answers and brainstorming. Here are a few points, so far, to summarize:

- Clear need for adding strings and files manipulation, access to databases, UI, ...
- Embeddable: inside a real Ada program? Yes, it is possible.

Here I just simplify what you find in hax.adb:

```
 ...
  CD : Compiler_Data;
 begin
  Set_Source_Stream (CD, [Some stream
access], [Possibly related file name]);
  Compile (CD);
  if CD.Err_Count = 0 then
    Interpret_on_Current_IO (CD);
  end if;

  …
```

- Freezing and restoring the state of the VM, and its data: it is possible. I still need to wrap the "global" variables for the state of the VM interpreter into an object type, but it is doable: they are located in a subpackage that could be turned into a record type with not much more effort than hitting it with a magic wand ;-) .
- Jupyter: seems something to be considered!
- Euphoria: seems a very good source of inspiration!
- The nasty need for "with HAC_Pack; use HAC_Pack;"

Yes it is still required - until the point where support for packages will be implemented in HAC.

However, it is already possible to cross-compile, since there is a "real" package (spec + body) in pure Ada in

exm/special/hac_pack.ads,

exm/special/hac_pack.adb.

For instance, the HAC demos and tests can be built with GNAT through the exm/hac_exm.gpr and test/hac_test.gpt project files respectively.

That's the big difference between HAC and a classical scripting language system: in the classical setup, the frontier is permanently set between the slow/interpreted/dynamic-typing part and the fast/compiled/static-typing part. Typically people need to reprogram parts or all of their scripts as C++ inserts in order to have a decent performance. With HAC you have the option to switch to your preferred compiler, with native code generation and optimization options. For a mix of VM/native code, we could imagine options or pragmas that triage units to VM or native code compilation. Just thinking loud...

[...]

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Thu, 30 Apr 2020 10:44:19 +0200

> Thank you all for your answers and brainstorming.

1. Where are the modules? It should be possible to write a module for the script e.g. a package that has functions and procedures, which call to Ada implementations. I.e. calling Ada from the script.
2. What about exceptions handling, the ones propagating out of the script into the Ada caller?
3. Aborting the script. Ideally the Interpret_on_Current_IO you mentioned must be abortable per some event set via protected object, for example. E.g. from another task provided Interpret_on_Current_IO runs on the caller's context. The interpreter will look for the event periodically and propagate exception Canceled_Error if the event is set.

From: gautier_niouzes@hotmail.com
Date: Fri, 1 May 2020 00:31:15 -0700

> 1. Where are the modules? [...]

On the to-do list :-)

> 2. What about exceptions handling [...]

Same, but this is a lower-hanging fruit: the VM has error states that could be grouped into an exception_raised state, which would trigger the expected behaviour (with an exception identity and message). If the exception is not handled from the VM, the VM interpreter would raise Unhandled_HAC_Exception with a message.

> 3. Aborting the script. [...]

I add this to the to-do list right now, thanks!

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 1 May 2020 09:51:25 +0200

> [Previous message entire quote. --arm]

With these changes I would consider integrating it into here:

http://www.dmitry-kazakov.de/ada/max_home_automation.htm#5

I presume you have an equivalent of Unbounded_String, records and arrays of. That should make it. Is https://sourceforge.net/projects/hacadacompiler/ the major source where you publish updates?

From: gautier_niouzes@hotmail.com
Date: Fri, 1 May 2020 08:46:33 -0700

> With these changes I would consider integrating it into here:

>

> http://www.dmitry-kazakov.de/ada/max_home_automation.htm#5

>

> I presume you have an equivalent of Unbounded_String, records and arrays of. That should make it.

On top of the to-do list now :-). But I have an idea how to implement it...

> Is

>

> https://sourceforge.net/projects/hacadacompiler/

>

> the major source where you publish updates?

Yep. With a mirror @

https://github.com/zertovitch/hac

in case of allergies to SourceForge.

Getting the Three-Letter Time Zone Abbreviation

From: Bob Goddard <1963bib@googlemail.com>
Subject: Getting the 3 letter time zone abbreviation
Date: Wed, 29 Apr 2020 01:46:58 -0700
Newsgroups: comp.lang.ada

I'm sure this has been asked many times...

I need to get the 3 letter time zone abbreviation.

Does anyone have code that can do that?

Neither Ada.Calendar nor ada_util can get it.

I did take a look at the tz db source code, and I shuddered multiple times.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Wed, 29 Apr 2020 11:09:37 +0200

>

> I need to get the 3 letter time zone abbreviation.

3-4 you mean, e.g. CEST.

> Does anyone have code that can do that?

I had only a partial success. I used GTK/GLib time zone functions. The abbreviation of the zone name is the thing. Unfortunately it works poorly under Windows, and Windows updates tend to break time zone settings [*] I needed to plant various fallbacks to deduce the zone from UTC offset.

Anyway, Ada bindings are here:

http://www.dmitry-kazakov.de/ada/gtkada_contributions.htm#5.14

[*] I believe it was the case why one could not log into Origin account for a couple of days not so long ago. Their server verified the time zone and blocked access because Windows reported garbage.

From: Bob Goddard <1963bib@googlemail.com>
Date: Wed, 29 Apr 2020 12:20:13 -0700

> I used GTK/GLib time zone functions. [...]

Seems easier just to import strftime and call it requesting just "%Z". This is on Linux, but MS suggests it should also work on Windows.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Wed, 29 Apr 2020 21:53:08 +0200

> Seems easier just to import strftime and call it requesting just "%Z". This is on Linux, but MS suggests it should also work on Windows.

An interesting idea. Did you try it under Windows? (There is a suspicious remark that it depends on the setlocale)

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Thu, 30 Apr 2020 23:11:34 +0200

[...]

OK, I tested it. As expected it does not work. [...]

Windows POSIX layer relies on Windows API. If the API does something wrong, so would whatever POSIX function.

From: Bob Goddard <1963bib@googlemail.com>
Date: Sat, 2 May 2020 05:46:41 -0700

Here goes...

On Linux and the various BSD's, the tm structure has been extended to include:

```
long int tm_gmtoff;
/* Seconds east of UTC.  */

const char *tm_zone;
/* Timezone abbreviation.  */
```

When you make a call to localtime_r, these are filled in with the correct info, at least on Linux.

The other big iron Unix's do not have this extension, and neither does Windows.

[...]

From: Bob Goddard <1963bib@googlemail.com>
Date: Sat, 2 May 2020 12:25:16 -0700

[...]

Anyway, turns out Windows does not support the IANA 3/4 letter tz database.

Calling tzset and examining the returned array, returns a string similar to "New Zealand Daylight Time".

tzset, at least on Linux, and I assume every other unix type system which uses the tz database does returns the 3/4 letter tz name.

Ho hum!

The History Behind Natural'First = 0

From: reinert <reinkor@gmail.com>
Subject: What is the history behind Natural'First = 0?
Date: Thu, 30 Apr 2020 21:51:07 -0700
Newsgroups: comp.lang.ada

I have been wondering about this for years:

Why Natural'First = 0?

There is no consensus about including 0 among the natural numbers. Since there is a Positive (Positive'First = 1), one may expect Natural'First = 0. Except for this, I find little intuition in "Natural'First = 0".

Copy form: https://en.wikipedia.org/wiki/Natural_number#History

Some definitions, including the standard ISO 80000-2,[1][2] begin the natural numbers with 0, corresponding to the non-negative integers 0, 1, 2, 3, …, whereas others start with 1, corresponding to the positive integers 1, 2, 3, …,[3][4] while others acknowledge both definitions.[5] Texts that exclude zero from the natural numbers sometimes refer to the natural numbers together with zero as the whole numbers, but in other writings, that term is used instead for the integers (including negative integers).[6]

Is the key point here: "the standard ISO 80000-2" ?

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Fri, 1 May 2020 09:52:04 +0200

> Why Natural'First = 0 ?

Because that's the way it is ;-)

Anyway, type Integer is not the mathematical notion of (infinite) integers, and more generally computer types are only reduced abstractions of mathematical notions.

There is a need for a subtype of type Integer with lower bound 0, and another one for lower bound 1. The names have been chosen by Ichbiah following usual practice, they could have been anything else.

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Fri, 1 May 2020 01:38:12 -0700

[...]

Being a wiseacre, I'd like to point out RM 3.5.4(8):

"The set of values for a signed integer type is the (infinite) set of mathematical integers [, though only values of the base range of the type are fully supported for run-time operations]."

[...]

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Fri, 1 May 2020 12:24:15 +0200

> [...] The set of values for a signed integer type is the (infinite) set of mathematical integers [...]

Yes, but being even more pedantic, let me point out that Integer is not a type, it is a first named subtype.

The type is purely conceptual in Ada (wasn't like this in Ada83).

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Fri, 1 May 2020 11:14:19 -0700

[In reply to the original post. --arm]

The key here is that there has always been a terminology divide between North America and Europe (with UK & Canada sometimes going a 3rd way along British-Empire lines).

Generally, in the USA, the set of natural numbers is the set of positive integers, that is denoted $\mathbb{N}$ domestically or $\mathbb{N}$* when interacting with people outside of the USA to show the lack of zero. Generally, in the USA, the set of whole numbers is the set of positive integers, that is denoted either $\mathbb{N}_0$ or $\mathbb{Z}^+$.

Conversely, generally in UK and Europe, the set of natural numbers is the set of nonnegative integers, which is denoted $\mathbb{Z}$-$^+$. (The dispute even goes that far: having different double-struck/white Z mnemonic notation: $\mathbb{Z}^+$ versus $\mathbb{Z}$-$^+$.) Generally in UK and Europe, the set of counting numbers was formerly the set of positive integers, but the further away from the 19th century we get, whole numbers have at times become synonymous with the UK/European definition of natural numbers.

https://mathworld.wolfram.com/NaturalNumber.html

https://mathworld.wolfram.com/NonnegativeInteger.html

https://mathworld.wolfram.com/CountingNumber.html

https://en.wikipedia.org/wiki/Natural_number

Ichbiah showed his European culture by institutionalizing the European definition as the sole normative definition in Ada. (And don't even get me started on billion, trillion, and milliard.)

From: Robert A Duff <bobduff@TheWorld.com>
Date: Fri, 01 May 2020 17:36:40 -0400

[...]

I seem to recall an early version of Ada (or Green) that said "subtype Natural is Integer range 1..Integer'Last;". I could be misremembering that, and (if true) I don't remember what the 0..Integer'Last one was called.

Speaking of zero:

Q: What caused the fall of the Roman Empire?

A: They didn't know about zero, so they had no way to terminate the strings in their C programs. Har, har.

From: Keith Thompson <Keith.S.Thompson+u@gmail.com>
Date: Sun, 03 May 2020 13:08:37 -0700

> I seem to recall an early version of Ada (or Green) that said "subtype Natural is Integer range 1..Integer'Last;".

Yes, I remember that. I found a copy of the 1979 Preliminary Ada Reference Manual from SIGPLAN Notices, June 1979 at

https://dl.acm.org/doi/pdf/10.1145/956650.956651

The section covering package STANDARD says:

```
subtype NATURAL is INTEGER range
1 .. INTEGER'LAST;
type STRING is array (NATURAL) of
CHARACTER;
```

There was no predefined subtype starting at 0. I don't know just when NATURAL was changed to start at 1 and POSITIVE was introduced.

(And I'm glad they decided to stop using ALL_CAPS for identifiers).

> Speaking of zero:

>

> Q: What caused the fall of the Roman Empire?

> A: They didn't know about zero, so they had no way to terminate the strings in their C programs. Har, har.

But it wasn't all that bad, since they only had 100 programs.

A Centurion walks into a bar. He holds up two fingers. "Five beers, please."

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Mon, 04 May 2020 10:22:03 -0400

>The 1980 edition had the same thing. I know there was another preliminary version in 1982 (before the first official standard in 1983), but I don't know what it said.

What is your definition of "official standard"? MIL-STD-1815 and 1815A /are/ official standards -- just not international standards.

ANSI/MIL-STD-1815A (dated 22 Jan 1983, approved 17 Feb 1983 "superseding MIL-STD-1815 10 Dec 1980") has the 0-based NATURAL and 1-based POSITIVE.

Preconditions Rock

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: preconditions rock
Date: Sun, 24 May 2020 17:00:56 -0700
Newsgroups: comp.lang.ada

I just have to say that I'm really appreciating how much preconditions help during development; I'm getting way better error messages when tests fail, so it is very easy to fix the problems.

[...]

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Mon, 25 May 2020 10:24:21 +0200

> I just have to say that I'm really appreciating how much preconditions help during development; I'm getting way better error messages when tests fail, so it is very easy to fix the problems.

Certainly, but preconditions are a way of thinking during design to create correct software. It's nice to have them directly supported in a language, but they're nothing new. I used preconditions in Ada 83 over 30 years ago.

From: Anh Vo <anhvofrcaus@gmail.com>
Date: Mon, 25 May 2020 10:14:01 -0700

> [...] I used preconditions in Ada 83 over 30 years ago.

But now the precondition is elevated. In addition, the codes and comments are always in synchronization. This will be the music to the ears of maintainers.

Ada in Code Art

From: Marius Amado-Alves <amado.alves@gmail.com>
Subject: Ada in code art
Date: Tue, 26 May 2020 03:21:13 -0700
Newsgroups: comp.lang.ada

Strangely pleased to find Ada included in the 128-Language Uroboros Quine.

https://www.youtube.com/watch?v=6avJHaC3C2U&feature=youtu.be&t=2081

[The alluded program is compiled in a starting language (Ruby), and when run it outputs the source code for a new program in a new language (Rust). When compiled in this second language, it produces a program that when run outputs the source code of a program in a third language. After going through over a hundred similar operations, the original program in Ruby is emitted by its predecessor, in REXX. The languages are ordered alphabetically to boot. The source code is also obfuscated in the form of an Uroboros. Find the source code at [1]. --arm]

[1] https://github.com/mame/quine-relay

Multiline Strings in Ada

[A subthread on multiline strings evolved from the “Ada++” thread (found in the Ada in Jest section). --arm]

From: "Nasser M. Abbasi" <nma@12000.org>
Subject: Ada++
Date: Thu, 28 May 2020 23:38:16 -0500
Newsgroups: comp.lang.ada

> [...] fix things like string types.

Yes, for example Ada is one of few languages that still does not have multiline raw string support. (may be also Fortran)

https://en.wikipedia.org/wiki/Here_document

Check also

https://rosettacode.org/wiki/Here_document

"A here document (or "heredoc") is a way of specifying a text block, preserving the line breaks, indentation and other whitespace within the text."

See the Ada answer above

"Ada has neither heredocs nor multiline strings. A workaround is to use containers of strings:"

Python, Perl, ruby, even C++11 added multiline raw string and more languages.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Fri, 29 May 2020 08:06:31 +0200

> "Ada has neither heredocs nor multiline strings. A workaround is to use containers of strings:"

Please provide a use case showing how these features are necessary/important. [...]

From: fabien.chouteau@gmail.com
Date: Fri, 29 May 2020 02:23:20 -0700

> Please provide a use case showing how these features are necessary/important.

https://github.com/alire-project/alire/blob/dfa1e1e8029dee2959742b73ed8a0fc96e22c8de/src/alr/alr-commands-index.adb#L171

[This is a code fragment in which standard vector containers of String are used to store paragraphs of text. --arm]

From: raph.amiard@gmail.com
Date: Fri, 29 May 2020 02:43:47 -0700

> On Friday, May 29, 2020 at 8:06:34 AM UTC+2, J-P. Rosen wrote:

> > Please provide a use case showing how these features are necessary/important.

Or in GNAT: https://github.com/aosm/libstdcxx_SUPanWheat/blob/02812415a478d43bcc37a17bb779fcab146fbe4c/libstdcxx/gcc/ada/snames.adb#L59

Or in Libadalang:

https://github.com/AdaCore/libadalang/blob/1cf553d5fc37317c670888f0893bc25560c85b7b/ada/extensions/src/libadalang-env_hooks.adb#L47

Those are just two examples on the top of my mind. Every time you want to embed a multiline string in an Ada app you need to go through this frankly annoying gymnastics.

This is also annoying for compiler writers. In Libadalang we need to recognize those as special cases because they can create comb trees of unbounded depth.

They are totally useless. (Embedding was never a good idea, embedded SQL, embedded machine code etc.)

Demonstrably not. Anyway those kinds of blanket statements tend to be false in general, I hope you can see that there are many legitimate use cases for this, and that the fact that you did not need it doesn't mean it's useless.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Fri, 29 May 2020 22:57:50 +0200

> Or in Libadalang:

>

> https://github.com/AdaCore/libadalang/blob/1cf553d5fc37317c670888f0893bc25560c85b7b/ada/extensions/src/libadalang-env_hooks.adb#L47

This is a surprising example in its very principle. Is the specification of Standard hard-coded in Libadalang? This would mean that the definition of Integer et.alt. is not the one of the compiler you are using, but the one of Libadalang. Puzzled…

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 29 May 2020 13:00:16 +0200

>> [...] Every time you want to embed a multi line string in an Ada app you need to go through this frankly annoying gymnastics.

>

> A S/W engineer, when encountering "frankly annoying gymnastics" a 2nd time, creates an abstraction to hide the "frankly annoying gymnastics", and so never has to go through the "frankly annoying gymnastics" ever again.

Which BTW was, at least partially, done by AdaCore as the comment preceding the string constant in the cited source code reads:

"The content of the following string literal has been generated running GNAT with flag -gnatS, and then post-processed by hand."

How, e.g. images are supposed to be literally embedded in the code is beyond me. So for my projects I wrote a few lines Ada code generator that creates a nice Ada package per image. No need to read its content ever.

Even embedding text content is a non-starter beyond a few toy cases, because of formatting, markup, Unicode, fonts and thousands other issues. Literal text is pretty much a non-existent thing. Typewriter times are gone.

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Sat, 13 Jun 2020 04:40:53 -0500

>> "Ada has neither heredocs nor multiline strings. A

>> workaround is to use containers of strings:"

> Please provide a use case showing how these features are necessary/important.

I wanted some time ago to use Ada to generate a Latex file on the fly.

You might ask, why not use a Latex editor? Because this is different.

When writing a program to generate the Latex file, then one can do some computation in the program on the fly, and emit the resulting string into the Latex file as it is being composed.

This way each time the program is run, a new Latex file is generated, with possible new content each time.

This can be much faster/better than having to edit a static Latex file in the Latex editor and update the document manually each time new results are obtained for example, by manually copying some computation result from another program into the Latex document.

I do this all the time for example in Mathematica.

Each time I update something in the data, I run the program, which generates a brand new Latex file, then compile this Latex file to get the new PDF report. Much much faster than editing a Latex file each time something new changes.

But to do this, one has to be able to write, inside the Ada editor, as if one is using a plain Latex editor, and not worry about having to close strings every 80 characters or so and start a new line and having to append each string one by one. It is much better to write a large amount of text at once, and having its structure preserved as is.

This is what multi-line raw strings allow one to do.

It is like writing a program to generate a new program.

It is not possible to do this in Ada. Well, it is, but it will be very very cumbersome.

Here are some very basic examples using Ruby, Perl and C++

https://www.12000.org/my_notes/here_document/index.htm

To see an example using Mathematica, used to generate a web page, here is an example

https://mathematica.stackexchange.com/questions/152663/making-a-website-with-mathematica

it is the second answer there.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Sun, 14 Jun 2020 07:29:33 +0200

> I wanted some time ago to use Ada to generate a Latex file on the fly.

> [...]

I think it is a bad idea to have the template file in the code. Every time you change a coma in your text, you'll need to recompile your program.

A much better solution is to use AWS template parser. You keep the template separate from your program, and you can parameterize the content at will. By changing the template, you can even move from Latex to whatever-is-in-fashion-today without changing your program.

Mathematics Libraries, Big_Numbers support in Ada 202x

[Although it does not seem to include the functions requested in this topic, the Mathpaqs library by Gautier de Montmollin is a referent in Ada. https://mathpaqs.sourceforge.io/ --arm]

From: reinert <reinkor@gmail.com>
Subject: Any good package for mathematical function in Ada?
Date: Sun, 31 May 2020 03:46:46 -0700
Newsgroups: comp.lang.ada

Hello,

I would like to use for example the Bessel function from Ada. I need to program it from scratch?

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Sun, 31 May 2020 13:26:38 +0200

> I would like to use for example the Bessel function from Ada.

Do you mean something concrete? I don't remember how many dozens of Bessel functions and integrals exist. Do you need all of them?

> I need to program it from scratch?

That depends on the approximation method, of which there are lots. Something universal, well, if you have coefficients of Chebyshev series e.g. from

https://www.amazon.com/Special-Functions-Their-Approximations/dp/0124110371

I have an Ada implementation for:

http://www.dmitry-kazakov.de/ada/components.htm#15.2

> Any hint?

If you need something concrete and frequently used, you certainly could find a library and use it. Ada bindings to a mathematical library is a matter of minutes to do.

From: Jerry <list_email@icloud.com>
Date: Sun, 31 May 2020 16:25:31 -0700

> I would like to use for example the bessel function from Ada. I need to program it from scratch?

You have hit on what, for me, is a PITA for Ada, generally speaking, and that is a lack of broad numerical functions.

You can link to the GNU Scientific Library (GSL) or the Octave binary library (I have also once written a very simple special-case Octave code generator to run interpreter code since some functionality is not available as compiled code). You might also enjoy ALGLIB, IMSL, NAG, and NetLib depending on the licenses. You might be able to link to the Python libraries scipy and numpy if, as I suppose, they are written in C. Consult this list if you like:

https://en.wikipedia.org/wiki/List_of_numerical_libraries

BTW if you have access to Numerical Recipes it can be a lifesaver sometimes. I have an early book with Pascal code in an appendix.

If the binary you are linking to is in C or Fortran your job as an Ada programmer isn't too bad but if you have never done it, it will take you a while to figure it out. Just remember that Ada is built to do this. [...]

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 1 Jun 2020 12:19:14 +0200

> Anybody having a simple (complete - runnable) code example using GSL from Ada?

Assuming Windows.

Install MSYS2 if you did not already. [64-bit, GNAT GPL is not available for 32-bit anymore.]

Install GSL mingw-w64-x86_64-gsl under MSYS.

Now, assuming that MSYS2 is under C:\MSYS64\MinGW, here you go:

---gsl.gpr--------------

```
project GSL is
   for Main use ("test.adb");

   package Linker is
     for Default_Switches ("ada")
        use ("-L/c/msys64/mingw/lib", "-lgsl");
   end Linker;

end GSL;
```

---gsl.ads-------------

```
with Interfaces.C;  use Interfaces.C;

package GSL is
  function Bessel_J0 (X : double)
  return double;

private
   pragma Import (C, Bessel_J0,
   "gsl_sf_bessel_J0");
end GSL;
```

```
---test.adb----------->
with Ada.Text_IO;   use Ada.Text_IO;
with GSL;           use GSL;
with Interfaces.C;  use Interfaces.C;

procedure Test is
begin
   Put_Line ("J0(1)=" & double'Image
(Bessel_J0 (1.0)));
end Test;
```

The test produces:

J0(1)= 7.65197686557967E-01

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Mon, 1 Jun 2020 05:48:54 -0500

> The test produces:
>
> J0(1)= 7.65197686557967E-01

That is good. In Mathematica

N[BesselJ[0, 1], 100]

0.7651976865579665514497175261026632209092742897553252418615475491192789122152724401671806000989156340

[...]

Want 1,000 digits? 2,000 digits? all can be done.

I think these systems both link to GMP "GNU Multiple Precision Arithmetic Library" for this. "There are no practical limits to the precision " https://en.wikipedia.org/wiki/GNU_Multiple_Precision_Arithmetic_Library

https://www.wolfram.com/legal/third-party-licenses/gmp.html

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 1 Jun 2020 13:34:59 +0200

>
> Want 1,000 digits? 2,000 digits? all can be done.

[...]

As for GMP specifically, I think that arbitrary precision numeric types must be an integral part of Ada. Unfortunately, this would introduce the same mess Unbounded_String did. So, for now, I would not push for them until the language type system matures to accommodate them smoothly.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 5 Jun 2020 17:49:45 -0500

> As for GMP specifically, I think that arbitrary precision numeric types must be an integral part of Ada. Unfortunately, this would introduce the same mess Unbounded_String did. So, for now, I would not push for them until the language type system matures to accommodate them smoothly.

You're a little late for that. See Big_Integer:

http://www.ada-auth.org/standards/2xrm/html/RM-A-5-6.html

and Big_Real:

http://www.ada-auth.org/standards/2xrm/html/RM-A-5-7.html

Since Ada 202x has user-defined literals and user-defined Image, this is almost as good as a built-in number. The only downside would be using them in generics (like GEF), but most of those implementations assume a maximum precision that's not true for these so they'd need rewriting anyway.

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Sat, 6 Jun 2020 06:58:51 -0700

[...]

GNAT CE 2020 has them (albeit not in the latest RM form). They behave much like numbers.

```
pragma Warnings (Off);
with
Ada.Numerics.Big_Numbers.Big_Integers,
Ada.Numerics.Big_Numbers.Big_Reals;
use
Ada.Numerics.Big_Numbers.Big_Integers,
Ada.Numerics.Big_Numbers.Big_Reals;
pragma Warnings (On);
with Ada.Text_IO;
use  Ada.Text_IO;

procedure Main is
  I: Big_Integer := From_String ("   42   ");
  J: Big_Integer := 42;
  R: Big_Real := From_String("10.0")**100
   - 1.0;
  S: Big_Real := 10.0**100 - 1/1;
  D: Big_Real := 1 / 3;

begin

  Put_Line (Boolean'(I=J)'Image);
  Put_Line (to_String (R));
  Put_Line (Boolean'Image(R=S));
  Put_Line (to_String (Numerator (S)) &
  to_String (Denominator (S)));
  Put_Line (to_String (D, Aft => 110));
  Put_Line (to_String (Numerator (D)) &
  to_String (Denominator (D)));

end Main;
```

Deprecation of Xref Information

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: GNAT gcc flag for generating xref info?
Date: Wed, 3 Jun 2020 16:56:08 -0700
Newsgroups: comp.lang.ada

There used to be a flag -fdumpxref for the GNAT gcc C compiler that output .gli files, which gnatcoll.xref parsed just as it does .ali files for Ada code.

But that is apparently gone;

gcc.exe: error: unrecognized command line option '-fdumpxref'

gnat/.../cc1.exe --help shows a flag -fxref, but that is also gone:

gcc.exe: warning: switch '-fxref' is no longer supported

So how do I use gnatcoll.xref with C code?

From: Simon Wright <simon@pushface.org>
Date: Thu, 04 Jun 2020 17:43:59 +0100

You would have thought, given the maturity of GCC (:-)) that the ChangeLogs would contain some references to these switches. The only one I can see is that references to -fdump-xref in the Ada documentation were removed (because the switch was deprecated) in December 2017.

The current GNAT Studio documentation says that the gnatinspect cross-reference database is deprecated.

So I guess they're now using libadalang (and a putative libclang???) but since GNAT Studio isn't provided in the Mac CE 2020 pack and the 2019 version works fine on the rare occasions I need it I can't comment further.

From: Simon Wright <simon@pushface.org>
Date: Thu, 04 Jun 2020 21:15:50 +0100

> a putative libclang

In the Github sources! https://github.com/AdaCore/gps/tree/master/libclang

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 5 Jun 2020 03:47:12 -0700

> So I guess they're now using libadalang (and a putative libclang???)

There is https://github.com/cquery-project/cquery; I'll try that with eglot [a client for Language Server Protocol servers].

Putting Data in the .data Section

From: "Luke A. Guest" <laguest@archeia.com>
Subject: How can I get this data into the .data section of the binary?
Date: Tue, 16 Jun 2020 12:31:26 +0100
Newsgroups: comp.lang.ada

I'm trying to get some static data tables into the data section rather than be elaborated at runtime. I can see no reason why this particular set of types, records and aggregates cannot go into the data section.

I've searched for use of pragma Static_Elaboration_Desired, but there is very little information.

[Huge source code section removed. I have inserted explanations to make the discussion self-contained without requiring the Ada examples. --arm]

From: "Luke A. Guest" <laguest@archeia.com>
Date: Tue, 16 Jun 2020 14:03:39 +0100

[After suggestions to try pragmas Shared_Passive or Preelaborate. --arm]

So, I re-applied an old patch from onox which adds in preelaborate everywhere and it still doesn't work. The pragma is in the package, the disassembly for the package shows the various objects being stored in .bss not .data and there is an elaboration procedure which initialised these objects in the .bss in the final applications.

From: "Luke A. Guest" <laguest@archeia.com>
Date: Tue, 16 Jun 2020 14:44:02 +0100

>

> I understand your problem, but this is a compiler issue, not a language issue. There is no such thing as a "data section" in a high level, machine independent, definition of a programming language...

According to 10.2.1 it should be possible:

“is important that programs be able to declare data structures that are link-time initialized with aggregates, string_literals, and concatenations thereof.” etc.

Even adding pragma Preelaborable_Initialization (x) for each of the types, doesn't do anything.

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue, 16 Jun 2020 21:19:43 +0300

>

> I'm trying to get some static data tables into the data section [...]

I haven't tried with your example, but is GNAT specific pragma Linker_Section acceptable?

https://docs.adacore.com/gnat_rm-docs/html/gnat_rm/gnat_rm/implementation_defined_pragmas.html#pragma-linker-section

I use that for some AVR-Ada code when I want to relocate some of the stuff to progmem. [...]

From: "Luke A. Guest" <laguest@archeia.com>
Date: Wed, 17 Jun 2020 13:37:05 +0100

>

> I haven't tried with your example, but is GNAT specific pragma Linker_Section acceptable?

I can't really see how it would be as it would still require the compiler to actually generate the correct value in .data or .rodata space rather than a 0x0 which gets filled in later by elaboration.

Interesting though.

I wrote the above before trying. Trying to place it in ".rodata..." caused a compiler error, placing it in .data worked, but there is still elaboration code. Although changing the section caused many more references to the object for some reason.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Wed, 17 Jun 2020 21:55:04 -0500

> According to 10.2.1 it should be possible:

>

> is important that programs be able to eclare data structures that are link-time initialized with aggregates, string_literals, and concatenations thereof. etc.

>

> Even adding pragma Preelaborable_Initialization (x) for each of the types, doesn't do anything.

The requirement in the Ada Standard is that one should preelaborate data if the containing package is preelaborated. There's no requirement otherwise, and since it is an annex C requirement, it is not required of all Ada compilers. [...]

Anyway, I would expect a compiler to do it if it is possible. But it very often isn't possible for one reason or another [...].

The better question is why do you care? One ought to be concerned about whether performance is good enough for your application, and it's highly unlikely that the load time would have any impact on that whatsoever. [...] (The situation can be different on a bare machine, of course.)

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 18 Jun 2020 12:55:47 +0300

>

> The better question is why do you care? [...] (The situation can be different on a bare machine, of course.)

[...] here are my reasons for needing them, just for the record.

For background, both cases occurred in bare-machine systems in which the entire SW is stored in EEPROM and is then entirely copied to RAM for execution, _including_ the code and read-only (constant) data.

For my case of the large constant array, we needed to save RAM space, and did not want to spend RAM _both_ for the elaboration code that initialized the array (larger than the array itself) and for the array. [...]

For my case of the constant version-identifier string, the customer required the executable SW image (in EEPROM) to contain a version identifier at a fixed address. This is a very common requirement in this domain. Of course it can be implemented in many ways (directly in the linker command script, for example), but I was pleased to be able to do it in Ada with the Linker_Section pragma.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Sat, 20 Jun 2020 22:55:37 -0500

>...

> For my case of the large constant array, we needed to save RAM space, and did not want to spend RAM _both_ for the elaboration code that initialized the array (larger than the array itself) and for the array.

I suppose it would depend on the declaration of the array, but I would not expect that to be the case most of the time. Typically, one can initialize most Ada data types with a block-copy, which would only be a handful of bytes on most target machines. Of course, if you have lots of controlled types and tasks, you'd have issues, but those aren't preelaborable anyway (some code would need to be executed for them).

[...]

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sun, 21 Jun 2020 09:55:16 +0300

> I suppose it would depend on the declaration of the array, but I would not expect that to be the case most of the time. [...]

I'm sure you are right, most of the time.

In our case, however, even if the compiler would have done a block-copy, the *source* of the block-copy would also have been in RAM because the *whole* SW image was copied at boot from EEPROM to RAM (as is ESA practice). So the RAM consumption of the array would still be at least twice the array size.

A block copy from EEPROM to RAM would have been ok, in principle, but in our case the compiler/linker knew nothing about the program's EEPROM residence. That was Boot SW business.

And if the compiler could have created the static source data for a block copy, it could as well have placed the whole load-time initialized array (a constant) in the read-only-data segment, which was what we wanted, and got in the end.

This was in the days when RAM in ESA on-board computers was expensive static RAM; nowadays it is usually dynamic RAM, I believe, and typical RAM size has gone up by one or two orders of magnitude.

Ada on Apple's New Processors, Licensing Concerns

From: Jerry <list_email@icloud.com>
Subject: Ada on Apple's new procesors
Date: Mon, 22 Jun 2020 15:53:00 -0700
Newsgroups: comp.lang.ada

Apple is beginning its third nightmare transition to a new processor family. What does this mean for Ada on macOS?

Can we hope for a native compiler anytime soon? We will have Rosetta 2 until we don't. (Original Rosetta lasted for two OS generations and then it was taken away.) I could tell you the story of needing to run a small PowerPC program to set up a slightly old Apple WiFi device a couple years ago. Buy Parallels. Call Apple and send $30 to get Snow Leopard Server--that's 10.6. Virtualize Snow Leopard Server on Parallels to run the WiFi set-up program in Rosetta.)

From: Vadim Godunko <vgodunko@gmail.com>
Date: Tue, 23 Jun 2020 03:42:46 -0700

> Apple is beginning its third nightmare transition to a new processor family. What does this mean for Ada on macOS?

>

> Can we hope for a native compiler anytime soon?

I suppose a native toolchain will be based on LLVM, thus it will allow to use GNAT LLVM on new processors.

[A large discussion is omitted at this point on the implications of GCC code generation in regard to the Runtime Library Exception (RLE) clause of GPLv3. However, as later was pointed out, GNAT LLVM does not rely on GCC. Yet, the bitcode of LLVM might still clash with the RLE and/or Apple Store terms of use. The discussion is ongoing and will be included as a whole in the next News Digest. --arm]

Ada in Jest

[I believe this project to be firmly tongue in cheek. But, who knows, and it sparks looots of discussion, some of it valuable... --arm]

Ada++

From: Jerry <list_email@icloud.com>
Subject: Ada++
Date: Thu, 28 May 2020 15:33:14 -0700
Newsgroups: comp.lang.ada

Ada++. YABL? Please discuss.

http://www.adapplang.com/

[Ada++ could be summarized as Ada with curly braces. See the "Hello, World!" example that follows, taken from the website. --arm]

```
use Ada.Wide_Text_IO;
proc Main:
{
  Put_Line ("Hello World");
}
```

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Thu, 28 May 2020 21:09:48 -0500

> Ada++. YABL? Please discuss.

> http://www.adapplang.com/

I do not know if this is real or just a joke.

But I do not like Ada++. I actually prefer the Pascal type constructs which Ada uses, which is explicit "Begin" "End" and "If" "Then" "Else", "LOOP", etc...

I do not like brackets {}. I find Pascal constructs more algorithmic and makes the code and the logic more clear.

btw, I do not think changing Ada syntax to make it look like C and C++ is the solution to making "Ada" become more popular. If so, then they should change Ada to make it use Python syntax in that case :)

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Thu, 28 May 2020 20:45:46 -0700

> Ada++. YABL? Please discuss.

> http://www.adapplang.com/

[...]

In mild support of their efforts, I would suggest that the Ada++ team go digging deep into old SIGADA and Tri-Ada academic papers at dl.acm.org [...] The article below is the biggest inventory of alternate variants of Ada that were discarded as proposal Green morphed into mil-standard Ada post-Steelman. [...] What a next-gen Ada would fix [...] Hint: more radical semantic maturations [...] greater amounts of orthogonality such as constant members of records as required in Steelman 3-3F [...] resurrection of the old a.app Ada-interpreter-within-the-Ada-compiler that was in DEC/Sun Ada compilers [...] then drastically extending a.app's capabilities for multistage programming beyond OCaml-P4's.

https://dl.acm.org/doi/pdf/10.1145/989791.989792

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Fri, 29 May 2020 08:41:32 -0700

> [...]

> https://dl.acm.org/doi/pdf/10.1145/989791.989792

In addition to the prose summary linked above, here is a much more fine-grained list of when features of Green or Ada were added or taken away. It is available without cost until 30 June 2020. Ada++ could revisit a great multitude of these decisions, some of which were from design-by-committee HOLWG reviewers not from Ichbiah's mastermind vision/intent.

https://dl.acm.org/doi/pdf/10.1145/24611.24614

From: cantanima.perry@gmail.com
Date: Thu, 28 May 2020 20:54:41 -0700

> Ada++. YABL? Please discuss.

> http://www.adapplang.com/

Didn't AdaCore have an April Fool's Joke to this effect?

[A subthread on beginner shock starts here. --arm]

From: Rick Newbie <nuttin@nuttn.nowhere>
Date: Thu, 28 May 2020 19:57:34 -0700

> btw, I do not think changing Ada syntax to make it look like C and C++ is the solution to making "Ada" become more popular. If so, then they should change Ada to make it use Python syntax in that case :)

As an Ada newbie I can tell you what is the most off-turning thing about Ada:

1) The IDE does not measure up to Visual Studio

2) The GDB debugger

3) The fact that you need lots of various external libraries and you have to deal with the Linux hell of library versions and installation to accomplish basic things like a graphical user interface.

The curly braces are definitely not a problem

From: raph.amiard@gmail.com
Date: Fri, 29 May 2020 02:49:08 -0700

> > btw, I do not think changing Ada syntax to make it look like C and C++

> > is the solution to making "Ada" become more popular. If so, then

> > they should change Ada to make it use Python syntax in that case :)

>

> As an Ada newbie I can tell you what is the most offturning thing about

> Ada:

> 1) The IDE does not measure up to Visual Studio

> 2) The GDB debugger

What might be of interest to you is using VS Code for Ada programming. AdaCore already provides an extension, and VSCode's front-end on top of GDB is pretty neat :)

> 3) The fact that you need lots of various external libraries and you have to deal with the Linux hell of library versions and installation to accomplish basic things like a graphical user interface.

The Ada library ecosystem is certainly lacking. Hopefully Alire [a package manager] will help with that situation in the long run.

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Fri, 29 May 2020 13:09:18 +0200

> The curly braces are definitely not a problem

If you ever lost one in the middle, you know that they ARE a problem

end if/end case /end proc-name /end func.name /end package name /end loop etc makes it much easier to see where something ends

```
switch (a)
{
  case 1 :
   {
     if (B==C)
     {
        while (true)
        {
          doSometing1();
          done = doSometing2();
          if done break;
        }
     }
  }
}
```

compared to

```
case a is
  when 1 =>
    if B = C then
      loop
        Do_Someting1;
        Done := Do_Someting2;
        exit when Done;
      end loop;
    end if;
  when others => null;
end case;
```

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Fri, 29 May 2020 14:57:14 -0700

> If you ever lost one in the middle, you know that they ARE a problem

> [...]

Well, to be fair, C and its progeny have botched the original BCPL block-statement bracketing hints.

https://dl.acm.org/doi/pdf/10.1145/988131.988138

As depicted in the article above, the BCPL feature applied to C would be the following, including compile-time enforcement to assure that the programmer-chosen bra tags match the programmer-chosen ket tags (where bra-ket is slang for open bracket and close bracket).

```
switch (a)
{sa
    case 1 :
     {c1
       if (B==C)
       {ifBC
          while (true)
          {wT
            doSometing1();
            done = doSometing2();
            if done break;
          }wT
       }ifBC
    }c1
}sa
```

As also mentioned in the above-linked article, PL/I had an analogous label-based mechanism for LBL: block-statement here END LBL; which, btw, Green-Ada partially borrowed due its limited amount of competitiveness with Red-PL/I.

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Sun, 31 May 2020 13:40:08 +0200

> Well, to be fair, C and its progeny have botched the original BCPL block-statement bracketing hints.

Meaning they do not have them?

Which is something I find a bad thing - or if you turn it around - I find having it a really good thing

I also noted some years ago that end procedure-name/end function-name is not mandatory in Ada. However GNAT has a style option to warn if they are missing. That is a good thing too. Other compilers may have that - but I don't know.

I also find lone begin/end - as in Pascal - just as hard to read. And difficult to find if you happen to lose one - or pasting code into another piece of code.

From: Rick Newbie <nuttin@nuttn.nowhere>
Date: Fri, 29 May 2020 18:36:57 -0700

>> The curly braces are definitely not a problem

OK I formulated it the wrong way. What I meant was that having curly braces or not was not the problem. I should have said "Begin/End is not the problem"

From: ric.wai88@gmail.com
Date: Sat, 30 May 2020 08:25:54 -0700

Guys, this was an April Fools Day joke.

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Sat, 30 May 2020 13:56:45 -0700

> Guys, this was an April Fools Day joke.

… perhaps by the same Ada++ name elsewhere, but this Ada++ seems to not be that one at all.

In addition to Stéphane Rivière's April 3rd (not April 1st, as one might expect for an April Fool's joke) dates of GitHub activity, the private-to-GoDaddy WHOIS information for adapplang dot com shows that the domain was registered on 21 January 2020 (not 31 March or 01 April, as one might expect for an April Fool's joke).

https://www.whois.com/whois/adapplang.com

Conversely, Ada++'s roadmap below seems blandly modest & mundane, referring only to work in-queue elsewhere: so far nothing that AdaCore itself wouldn't eventually do for GNAT. Therefore as an April Fool's joke, it isn't outlandish at all. Hence, where is the April-Fools humor in that?

http://www.AdappLang.com/docs.html

From: ric.wai88@gmail.com
Date: Sat, 30 May 2020 14:58:34 -0700

The original author posted it to Ada Comment on April 1st. Yes it is a fun side-project for him I'm sure, but it is not a serious thing (thankfully).

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 5 Jun 2020 17:37:43 -0500

> The original author posted it to Ada Comment on April 1st. Yes it is a fun side-project for him I'm sure, but it is not a serious thing (thankfully).

Which unfortunately got delayed several days because of me not being in the office to approve it, thus clobbering the joke.

As far as starting it early, I wrote AI12-0841-1 right after the previous year's April Fools day. And I spent time on it and the associated "news" post for a week before posting it. (And I failed to get Wordpress to post it on April 1st. Perhaps there is a pattern here. ;-) Doing these things right takes a decent investment of time.

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: "(c)" = event cancelled, and "(v)" = event is fully or partially held online.

2020

July 2 (c)(v)
23rd **European Joint Conferences on Theory and Practice of Software** (ETAPS'2020). Dublin, Ireland. ETAPS'2020 was first postponed from 25-30 April 2020 to fall 2020, but then cancelled and replaced by a 3-hour virtual online ETAPS 2020 event on 2 July 2020.

July 06-11 (v)
42nd **International Conference on Software Engineering** (ICSE'2020). Seoul, South Korea. ICSE'2020 was postponed from 23-27 May to 6-11 July (core) and 26 June - 19 July (co-located events). Topics include: the full spectrum of Software Engineering.

- June 28-30 — 3rd **International Conference on Technical Debt** (TechDebt'2020). Topics include: the business case for technical debt management; understanding causes and effects of technical debt; technical debt management within software life-cycle management; technical debt in design and architecture; technical debt and software evolution, maintenance, and aging; concrete practices and tools used to manage technical debt; debt remediation and refactoring; technical debt and quality attributes, such as security (especially at run-time); technical debt in (ultra-) large-scale systems, ecosystems, platforms and product lines; success and failure stories of technical debt management; education and training related to technical debt; etc.
- July 07-10 — **Software Engineering Education and Training** (SEET'2020). Topics include: novel methods of teaching software engineering skills, empirical studies describing software engineering education contexts, novel learning technologies that support software engineering education and training, well-substantiated arguments about what skills are most essential to learn, etc.
- July 13th — 8th **International Conference on Formal Methods in Software Engineering** (FormaliSE'2020). Topics include: approaches and tools for verification and validation; application of formal methods to specific domains, e.g. autonomous, cyber-physical, and IoT systems; scalability of formal methods applications; integration of formal methods within the software development lifecycle formal specification; model-based engineering approaches; formal methods in a certification context; formal approaches for safety and security-related issues; usability of formal methods; guidelines to use formal methods in practice; case studies developed/analyzed with formal approaches; experience reports on the application of formal methods to real-world problems; etc.

July 07-10 (v)
32nd **Euromicro Conference on Real-Time Systems** (ECRTS'2020). Modena, Italy.

- July 07 — 5th **Workshop on Security and Dependability of Critical Embedded Real-Time Systems** (CERTS'2020). Topics include: the intersection of security and dependability of embedded and real-time systems, with an emphasis on criticality and distribution.

July 13-15 (v)
28th **IEEE/ACM International Conference on Program Comprehension** (ICPC'2020), Seoul, South Korea. Co-located with ICSE'2020. ICPC'2020 was first postponed from 23-24 May to 10-11 October, and later moved to a virtual event format on 13-15 July. Topics include: tool support for program

	comprehension; comprehension of specific types of software systems, such as open/closed source, mobile applications, spreadsheets, web-based systems, legacy systems, product lines, libraries, multi-threaded applications, and systems of systems; comprehension in the context of diverse software process models and specific lifecycle activities, such as: maintenance, evolution, reengineering, migration, security, auditing, and testing; empirical evaluations of program comprehension tools, techniques, and approaches; human aspects in program comprehension; etc.
July 13-17 (v)	44th **Annual** IEEE **Conference on Computers, Software and Applications** (COMPSAC'2020). Madrid, Spain.
July 13-17 (v)	14th ACM/IFIP **International Conference on Distributed Event-Based Systems** (DEBS'2020). Montreal, Quebec, Canada. Topics include: systems dealing with collecting, detecting, processing and responding to events through distributed middleware and applications; embedded systems, real-time analytics, complex event processing, distributed programming, security, reliability and resilience, Internet-of-Things, cyber-physical systems, etc.
☺ August 19-21	26th IEEE **International Conference on Embedded Real-Time Computing Systems and Applications** (RTCSA'2020). Gangnueng, South Korea. Topics include: real-time scheduling, timing analysis, programming languages and run-time systems, middleware systems, applications and case studies of IoT and CPS, cyber-physical co-design, multi-core embedded systems, fault tolerance and security, etc.
August 26-28 (v)	46th **Euromicro Conference on Software Engineering and Advanced Applications** (SEAA'2020). Portoroz, Slovenia. Topics include: information technology for software-intensive systems; conference tracks on Embedded Systems and the Internet of Things (ES-IoT), Software Process and Product Improvement (SPPI), Model-Driven Engineering and Modeling Language (MDEML), etc.; special sessions on Cyber-Physical Systems (CPS), Software Engineering and Technical Debt (SEaTeD), etc.
September 01-04 (v)	31st **International Conference on Concurrency Theory** (CONCUR'2020). Vienna, Austria. Topics include: semantics, logics, verification and analysis of concurrent systems; basic models of concurrency; verification and analysis techniques for concurrent systems such as abstract interpretation, atomicity checking, model checking, race detection, run-time verification, static analysis, testing, theorem proving, type systems, security analysis, ...; distributed algorithms and data structures; theoretical foundations of architectures, execution environments, and software development for concurrent systems such as multiprocessor and multi-core architectures, compilers and tools for concurrent programming, programming models such as component-based, object-oriented, ...; etc.
☺ September 02-03 (v)	25th **International Conference on Formal Methods for Industrial Critical Systems** (FMICS'2020). Vienna, Austria. Co-located with CONCUR'2020 and FORMATS'2020. Topics include: case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; methods, techniques and tools to support automated analysis, certification, debugging, descriptions, learning, optimisation and transformation of complex, distributed, real-time, embedded, mobile and autonomous systems; verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability (e.g., scalability and usability issues); impact of the adoption of formal methods on the development process and associated costs; application of formal methods in standardisation and industrial forums.
September 06-09 (v)	15th **Federated Conference on Computer Science and Information Systems** (FedCSIS'2020). Sofia, Bulgaria. Event includes: Language Technologies and Applications (5th Workshop LTA'20), Scalable Computing (11th Workshop WSC'20), Cyber Security, Privacy and Trust (1st International Forum NEMESIS'20), Advances in Software and System Engineering (ASSE'20), Cyber-Physical Systems (7th Workshop IWCPS'20), Lean and Agile Software Development (4th International Conference LASD'20), Model Driven Approaches in System Development (6th Workshop MDASD'20), Software Engineering (40th IEEE Workshop SEW'20), etc. Deadline for submissions: July 3, 2020 (papers), July 17, 2020 (position papers).
September 08-11 (v)	13th **International Conference on the Quality of Information and Communications Technology** (QUATIC'2020). Faro, Portugal. Topics include: all quality aspects in ICT systems engineering and management; quality in ICT process, product and applications domains; practical studies; etc. Tracks on ICT verification and validation, safety, security and privacy, model-driven methods, agile methods, evolution in ICT / reengineering and refactoring, evidence-based software quality engineering, software quality education and training, etc.

September 14-18 (v)	18th **International Conference on Software Engineering and Formal Methods** (SEFM'2020). Amsterdam, the Netherlands. Topics include: software development methods (software evolution, maintenance, re-engineering, and reuse ...), design principles (programming languages, abstraction and refinement, ...), software testing, validation, and verification (model checking, theorem proving, and decision procedures; testing and runtime verification; other light-weight and scalable formal methods; ...), security and safety (security, privacy, and trust; safety-critical, fault-tolerant, and secure systems; software certification), applications and technology transfer (real-time, hybrid, and cyber-physical systems; education; ...), case studies, best practices, and experience reports.
September 15-18 (v)	39th **International Conference on Computer Safety, Reliability and Security** (Safecomp'2020). Lisbon, Portugal. Topics include: all aspects related to the development, assessment, operation and maintenance of safety-related and safety-critical computer systems; formal modelling, verification and validation; model-driven engineering; security and privacy protection mechanisms; safety/security co-engineering and risk assessment; testing, verification and validation methods & tools; qualification, assurance and certification methods & tools; cyber-physical threats and vulnerability analysis; safety and security guidelines, standards and certification; etc. Domains of application include: railways, automotive, space, avionics & process industries; highly automated and autonomous systems; telecommunication and networks; safety-related applications of smart systems and IoT; critical infrastructures; medical devices and healthcare; surveillance, defense, emergency & rescue; logistics, industrial automation, off-shore technology; education & training; etc. Deadline for submissions: July 3, 2020 (position papers). Deadline for early registration: July 20, 2020.
September 20-25 (v)	**Embedded Systems Week** 2020 (ESWEEK'2020). Hamburg, Germany. ESWEEK'2020 was first moved from October 11-16 in Shanghai, China, to September 20-25 in Hamburg, Germany, and later moved to a virtual event format. Deadline for submissions: July 13, 2020 (PhD Student Forum extended abstracts).

Sep 20-25	**International Conference on Compilers, Architecture, and Synthesis for Embedded Systems** (CASES'2020). 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); Trusted IoT Day; etc.
Sep 20-25	**International Conference on Hardware/Software Codesign and System Synthesis** (CODES+ISSS'2020). Topics include: system-level design, hardware/software co-design, modeling, analysis, and implementation of modern embedded and cyber-physical systems, from system-level specification and optimization to system synthesis of multi-processor hardware/software implementations.
Sep 20-25	ACM SIGBED **International Conference on Embedded Software** (EMSOFT'2020). 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 compose computation, networking, and physical dynamics.

September 21-24 (v)	39th **International Symposium on Reliable Distributed Systems** (SRDS'2020). Shanghai, China. Topics include: distributed systems design, development and evaluation, with emphasis on reliability, availability, safety, dependability, security, and real-time.
September 21-25 (v)	35th IEEE/ACM **International Conference on Automated Software Engineering** (ASE'2020). 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: July 3, 2020 (Doctoral Symposium), July 10, 2020 (journal-first papers, Industry Showcase track posters & proposals, Student Research Competition), August 1, 2020 (Late Breaking Results track).

September 22-24	19th **International Conference on Intelligent Software Methodologies, Tools and Techniques** (SOMET'2020). Kytakyushu, Japan. Topics include: state-of-art and new trends on software methodologies, tools and techniques; software methodologies, and tools for robust, reliable, non-fragile software design; software developments techniques and legacy systems; automatic software generation versus reuse, and legacy systems; software evolution techniques; Agile Software and Lean Methods; formal methods for software design; software maintenance; software security tools and techniques; formal techniques for software representation, software testing and validation; software reliability; Model Driven Development (DVD), code centric to model centric software engineering; etc.
October 06-09	20th **International Conference on Runtime Verification** (RV'2020). Los Angeles, California, USA. Topics include: monitoring and analysis of the 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.
October 24-28	13th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2020). Porto, Portugal. ICST'2020 was postponed from 23-27 March to 24-28 October. Topics include: manual testing practices and techniques, security testing, model based testing, test automation, static analysis and symbolic execution, formal verification and model checking, software reliability, testability and design, testing and development processes, testing in specific domains (such as embedded, concurrent, distributed, ..., and real-time systems), testing/debugging tools, empirical studies, experience reports, etc.
October 28-31	25th **International Conference on Engineering of Complex Computer Systems** (ICECCS'2020), Singapore. Topics include: all areas related to complex computer-based systems, including the causes of complexity and means of avoiding, controlling, or coping with complexity, such as verification and validation, security and privacy of complex systems, model-driven development, reverse engineering and refactoring, software architecture, design by contract, agile methods, safety-critical and fault-tolerant architectures, real-time and embedded systems, systems of systems, cyber-physical systems and Internet of Things (IoT), tools and tool integration, industrial case studies, etc.
November 08-13 (v)	28th ACM **Joint European Software Engineering Conference** and **Symposium on the Foundations of Software Engineering** (ESEC/FSE'2020). San Francisco, California, USA. Topics include: agile software development; component-based software engineering; configuration management and deployment; cyber physical systems; debugging; dependability, safety, and reliability; education; embedded software; emerging domains of software; empirical software engineering; formal methods; middleware, frameworks, and APIs; mining software engineering repositories; model-driven engineering; parallel, distributed, and concurrent systems; program analysis; program comprehension; program repair; programming languages; refactoring; reverse engineering; safety-critical systems; scientific computing; security, privacy and trust; software architecture; software economics and metrics; software evolution and maintenance; software modeling and design; software process; software product lines; software reuse; software testing; software visualization; specification and modeling languages; tools and environments; traceability; validation and verification; etc.
November 09-12 (v)	32nd **International Conference on Software Engineering Education and Training** (CSEET'2020). Munich, Germany. CSEET'2020 was first postponed from 28-31 July to 9-12 November, and later moved to a virtual event format. Topics include: Teaching formal methods (TFM), Teaching "real world" SE practices (TRW), Software quality assurance education (SQE), Global and distributed SE education (GDE), Open source in education (OSE), Cooperation between Industry and Academia (CIA), Training models in industry (TMI), Continuous education (CED), Methodological aspects of SE education (MAE), etc.
☺ November 15-16 (v)	ACM SIGAda's **High Integrity Language Technology Workshop on Safe Languages and Technologies for Structured and Efficient Parallel and Distributed/Cloud Computing** (HILT'2020), Chicago, Illinois, USA. Co-located with SPLASH 2020. Sponsored by ACM SIGAda. Topics include: practical use of High Integrity languages, technologies, and methodologies in construction of safe, structured, highly parallel and/or distributed/cloud applications; safe and productive languages and frameworks for development of structured parallel and/or distributed applications (e.g. Rust, Concurrent Collections, Ada 202X, Parsl); practical tools for applying static analysis and formal methods to parallel and/or distributed/cloud applications (e.g. SPARKProver, Java Pathfinder); etc. Deadline for submissions: September 4, 2020 (presentations, panels, full papers).

☺ November 15-17 (v)	34th **European Conference on Object-Oriented Programming** (ECOOP'2020). Berlin, Germany. ECOOP'2020 was moved from 13-17 July in Berlin, Germany, to 15-17 November in a virtual event format. Topics include: design, implementation, optimization, analysis, and theory of programs, programming languages, and programming environments.
☺ November 15-20 (v)	ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2020). Chicago, USA. Topics include: all aspects of software construction, at the intersection of programming, languages, and software engineering. Deadline for submissions: July 7, 2020 (DLS - Dynamic Languages Symposium), July 10, 2020 (SPLASH-E), July 15, 2020 (Doctoral Symposium), July 20, 2020 (GPCE abstracts - Generative Programming: Concepts & Experiences, SLE abstracts - Software Language Engineering), July 27, 2020 (GPCE papers - Generative Programming: Concepts & Experiences, SLE papers - Software Language Engineering), August 15, 2020 (Student Research Competition abstracts, Onward! 2nd round), September 1, 2020 (student volunteer applications), September 4, 2020 (PLMW Travel Grant Applications - PL Mentoring Workshop, workshop papers), September 10, 2020 (posters).
November 16-20 (v)	16th **International Conference on integrated Formal Methods** (iFM'2020), Lugano, Switzerland. Topics include: recent research advances in the development of integrated approaches to formal modelling and analysis; all aspects of the design of integrated techniques, including language design, verification and validation, automated tool support and the use of such techniques in software engineering practice.
November 16-20	23rd **Ibero-American Conference on Software Engineering** (CIbSE'2020). Curitiba, Brazil. CIbSE'2020 was postponed from 4-8 May to 16-20 November. Event includes Software Engineering Track (SET) and Experimental Software Engineering Track (ESELAW).
November 18-20 (v)	27th **Static Analysis Symposium** (SAS'2020), Chicago, Illinois, USA. Topics include: all aspects of static analysis, such as abstract interpretation, data flow analysis, debugging, emerging applications, model checking, program optimizations and transformations, program verification, security analysis, tool environments and architectures, type checking, etc.
November 25-27 (v)	21st **International Conference on Product-Focused Software Process Improvement** (PROFES'2020). 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 13, 2020 (full research paper abstracts), July 20, 2020 (full research papers), July 31, 2020 (short papers, industry papers, tutorials), September 14, 2020 (Journal-First papers).
November 25-27 (v)	23rd **Brazilian Symposium on Formal Methods** (SBMF'2020), Ouro Preto, MG, Brazil. Topics include: applications of formal methods to software development, code generation testing, maintenance, evolution, reuse, ...; specification and modelling languages (such as 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 (such as type systems, models of concurrency, security, ...); verification and validation (such as 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 10, 2020 (abstracts), July 17, 2020 (full papers).
Nov 29 - Dec 03 (v)	18th **Asian Symposium on Programming Languages and Systems** (APLAS'2020), Fukuoka, Japan. Topics include: design of languages and type systems; domain-specific languages; compilers, interpreters, abstract machines; program analysis, verification, model-checking; software security; concurrency and parallelism; tools and environments for programming and implementation; etc. Deadline for submissions: July 3, 2020 (submissions).
☺ December 01-04	41st IEEE **Real-Time Systems Symposium** (RTSS'2020). Houston, Texas, USA. Deadline for submissions: July 2, 2020 (papers), September 3, 2020 (Hot Topics Day event proposals), September 23, 2020 (brief presentations), October 15, 2020 (Hot Topics Day special sessions, tutorial drafts).
December 01-04	27th **Asia-Pacific Software Engineering Conference** (APSEC'2020), Singapore. Topics include: agile methodologies; component-based software engineering; configuration management and deployment; 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 10, 2020 (Technical Research papers, Software Engineering in Practice (SEIP) papers), August 14, 2020 (Early Research Achievements (ERA) abstracts), August 21, 2020 (ERA papers), October 2, 2020 (posters).

December 02-04 (v) — 19th **International Conference on Software Reuse** (ICSR'2020). Hammamet, Tunisia. ICSR'2020 was postponed from 9-11 November to 2-4 December, and moved to a virtual event format. Theme: "Reuse in emerging software engineering practices". Topics include: new and innovative research results and industrial experience reports dealing with all aspects of software reuse within the context of the modern software development landscape. Deadline for submissions: August 15, 2020 (paper abstracts), August 18, 2020 (full papers), October 2, 2020 (workshops, tutorials, Doctoral Symposium).

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

December 11-14 — 20th IEEE **International Conference on Software Quality, Reliability and Security** (QRS'2020), Macau, China. QRS'2020 was postponed from 27-31 July in Vilnius, Lithuania to 11-14 December in Macau, China. Topics include: reliability, security, availability, and safety of software systems; software testing, verification, and validation; program debugging and comprehension; fault tolerance for software reliability improvement; modeling, prediction, simulation, and evaluation; metrics, measurements, and analysis; software vulnerabilities; formal methods; operating system security and reliability; benchmark, tools, industrial applications, and empirical studies; etc.

2021

January 18-20 — 16th **International Conference on High Performance and Embedded Architecture and Compilation** (HiPEAC'2021). Budapest, Hungary. Topics include: computer architecture, programming models, compilers and operating systems for embedded and general-purpose systems. Deadline for submissions: July 10, 2020 (workshops, tutorials).

January 19-21 — 13th **Software Quality Days** (SWQD'2021), Vienna, Austria. Topics include: improvement of software development methods and processes, testing and quality assurance of software and software-intensive systems, domain specific quality issues (such as embedded, medical, automotive systems), novel trends in software quality, etc.

March 22-26 — 36th ACM **Symposium on Applied Computing** (SAC'2021), Gwangju, Korea.

- Mar 22-26 — **Software Verification and Testing Track** (SVT'2021). Topics include: new results in formal verification and testing, technologies to improve the usability of formal methods in software engineering, applications of mechanical verification to large scale software, model checking, correct by construction development, model-based testing, software testing, static and dynamic analysis, abstract interpretation, analysis methods for dependable systems, software certification and proof carrying code, fault diagnosis and debugging, verification and validation of large scale software systems, real world applications and case studies applying software testing and verification, etc. Deadline for submissions: September 15, 2020 (regular papers, student research competition research abstracts).
- Mar 22-26 — **Embedded Systems Track** (EMBS'2021). Topics include: the application of both novel and well-known techniques to the embedded systems development. Deadline for submissions: September 15, 2020 (full papers).

March 27 - April 1 — 24th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2021), Luxembourg, Luxembourg. 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), SV-COMP (the 10th Competition on Software Verification). Deadline for submissions: October 15, 2020 (papers).

April 12-16 (v)	14th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2021), Porto de Galinhas, Brazil. Topics include: manual testing practices and techniques, security testing, model based testing, test automation, static analysis and symbolic execution, formal verification and model checking, software reliability, testability and design, testing and development processes, testing in specific domains (such as embedded, concurrent, distributed, ..., and real-time systems), testing/debugging tools, empirical studies, experience reports, etc. Deadline for submissions: September 10, 2020 (workshops), October 5, 2020 (abstracts), October 12, 2020 (papers).
May 23-29	43rd **International Conference on Software Engineering** (CSE'2021), Madrid, Spain. Topics include: the full spectrum of Software Engineering, such as testing and analysis (software testing, program analysis, validation and verification, fault localization, formal methods, programming languages), empirical software engineering (mining software repositories, software ecosystems, ...), software evolution (evolution and maintenance, debugging, program comprehension, API design and evolution, configuration management, release engineering and DevOps, software reuse, refactoring, reverse engineering, ...), social aspects of software engineering (human aspects of software engineering, agile methods and software processes, software economics, ethics in software engineering, ...), requirements, modeling, and design (requirements engineering, modeling and model-driven engineering, software architecture and design, tools and environments, variability and product lines, parallel, distributed, and concurrent systems, ...), dependability (software security, privacy, reliability and safety, performance, embedded / cyber-physical systems, ...), etc. Deadline for submissions: August 28, 2020 (technical track), October 1, 2020 (IEEE TCSE Harlan Mills Award nominations).
♦ June 07-11	25th **Ada-Europe International Conference on Reliable Software Technologies** (AEiC 2021 aka Ada-Europe 2021). Santander, Spain. AEiC'2020 was postponed from 8-12 June 2020 to 7-11 June 2021. Sponsored by Ada-Europe.
October 10-15	**Embedded Systems Week** 2021 (ESWEEK'2021). Shanghai, China. The venues for ESWEEK 2020 and 2021 were swapped. ESWEEK 2020 will now be held in Hamburg, Germany from September 20-25, 2020, and ESWEEK 2021 will be held in Shanghai, China from October 10-15, 2021.
November 20-26	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. Deadline for submissions: April 30, 2021 (abstracts), May 6, 2021 (full papers).
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

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

25th Ada-Europe International Conference on
Reliable Software Technologies (AEiC 2021)
Santander, Spain, June 2021

The containment measures put in place on a world scale against the COVID-19 pandemic, which is affecting us all so severely, have taken precedence in everybody's personal and professional life. The consequent regulations issued by national governments and public health authorities in the regard of public gatherings, obliged us to cancel the 25th Ada-Europe International Conference (AEiC 2020), which was scheduled for 8-12 June 2020, in Santander, Spain.

Consequent to that obligation, the organizing committee and its principal sponsor Ada-Europe have made a number of decisions, of which we highlight:

- Santander, Spain, will be the venue of the next Ada-Europe International Conference on Reliable Software Technologies in June 2021.
- The venue originally earmarked for the 2021 edition of the AEiC, Ghent, Belgium, will be the prime candidate to host the 2022 edition of the conference.
- The processing of the Journal-track submissions continues seamlessly, with a view to ensuring prompt open-access publication to those that gain acceptance. The authors of the accepted papers will be invited to present their work at the 2021 conference.
- The authors of the Industrial Presentations and other contributions were invited to forward their work to the Ada User Journal for publication.

An updated Call for Contributions and new deadlines will be provided in due course.

We express our heartfelt friendship to all members of our community and their families in these difficult times, and look forward to the restoration of a safe normality.

See you all soon!

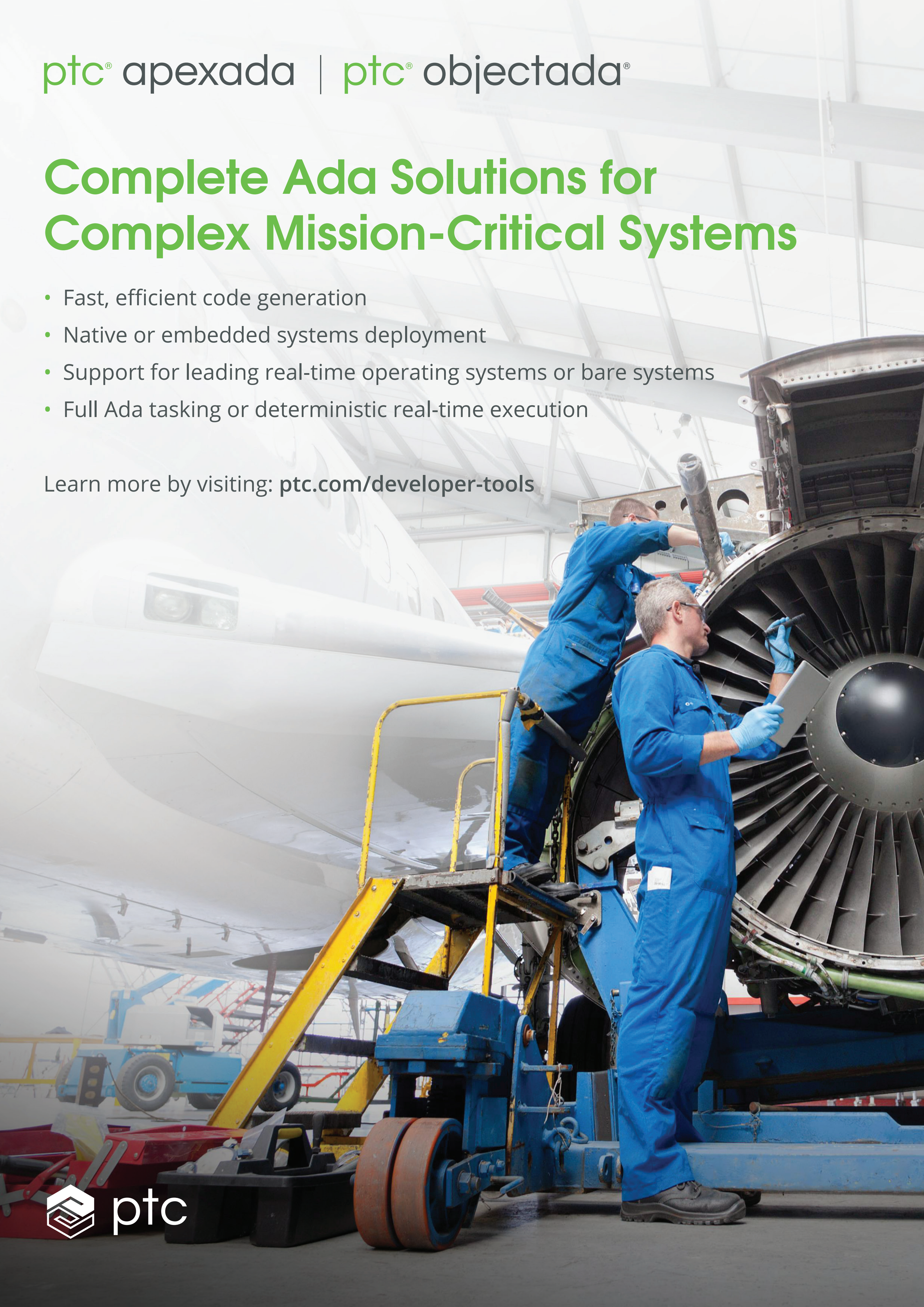
The Ada-Europe Board

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

Spunky, a Genode kernel in Ada/SPARK

Martin Stein

Genode Labs, Friedrichstraße 26, D-01067 Dresden; email: martin.stein@genode-labs.com

Abstract

This article introduces the Spunky microkernel project recently presented at the 10th Ada Developer Room at FOSDEM 2020 [3]. The Spunky project aims for providing a kernel for the Genode OS framework written in Ada that may subsequently be transferred to SPARK to enable automated proving of fundamental kernel aspects.

Keywords: Genode, Microkernel, Ada, SPARK.

1 Background

Since 2010 I'm involved in the development of the Genode OS framework at Genode Labs in Dresden (Germany). The Genode OS framework is a free open-source tool kit for building highly secure component-based operating systems. The resulting operating systems may scale from embedded devices to dynamic desktop systems [5] [4]. The development of Genode started in 2008 and is driven to this day by Genode Labs [2] together with a growing international Genode community.

During my time at Genode Labs I worked at topics like device emulation, an in-house microkernel, userland drivers, timing, networking, testing infrastructure, and, most recently, block encryption with SPARK. This latter project provided to me the opportunity to get into the languages Ada and SPARK and the idea behind them. Enthusiasm for this way of developing software grew inside me and I wanted to use it on a more fundamental part of my day-to-day working system, namely the kernel, to see what I can achieve.

Fortunately, one of the characteristics of Genode is its portability regarding the underlying kernel. It builds on a generic microkernel abstraction to such an extent that userland programs are actually kernel-agnostic. The framework already runs on a variety of microkernels like SeL4, NOVA, and Fiasco OC as well as on the Linux kernel and the Muen separation kernel. Furthermore, the project also features its own microkernel named "Base-HW" written in C++ like most of the Genode framework.

The Base-HW kernel has played a special role in the development of Spunky. The design of Base-HW puts a particular emphasis on a minimalistic feature set and a well structured code base. It consists of less than 10'000 lines of code while running on x86, ARM, and RISCV and it doesn't contain

Presented at the 10th Ada Developer Room at FOSDEM 2020, Brussels, 1 Feb 2020.

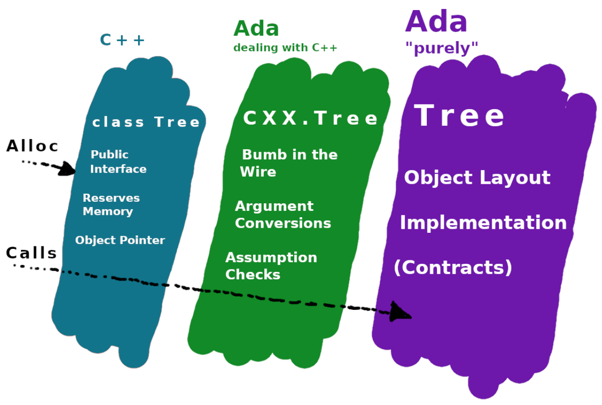


Figure 1: Choosen approach for interfacing Ada from C++

blocking or time-intensive code. In fact, it is a mere single-threaded state machine considering that kernel passes are very fast. Due to these properties, the Base-HW design seemed especially attractive for an implementation aiming for automated correctness proofs. In the context of the Spunky project, I'm trying to evaluate this assumption.

2 The Spunky approach

To achieve a smooth work flow with small implementation steps and high testability, a hybrid approach was choosen for the first project phase: Starting with Base-HW's C++ implementation, I sucessively replaced individual C++ classes with Ada packages as drop-in replacements while maintaining the binary interface of the original C++ classes intact. Packages of Spunky that are already finished are complemented by C++ implementations of the missing packages taken from Base-HW to form a complete kernel. This way, the resulting kernel gradually transitions to a self-standing, pure Ada implementation.

The interfacing between Ada and C++ is done as follows: As Figure 1 shows, an Ada package ("Tree") that shall be called from C++ is presented by a hollow class ("class Tree") in C++. The class declares methods that match the public sub-programs of the Ada package. But their implementation is only done in Ada. When calling such a method in C++ the 'this' pointer is interpreted as 'in' or 'in-out' parameter of a record type that is private to the package. The size of the C++ class is therefore inflated artificially to the size of this record.

In-between the C++ class and the Ada package, a transitional Ada package is interposed ("CXX.Tree"). It has the same

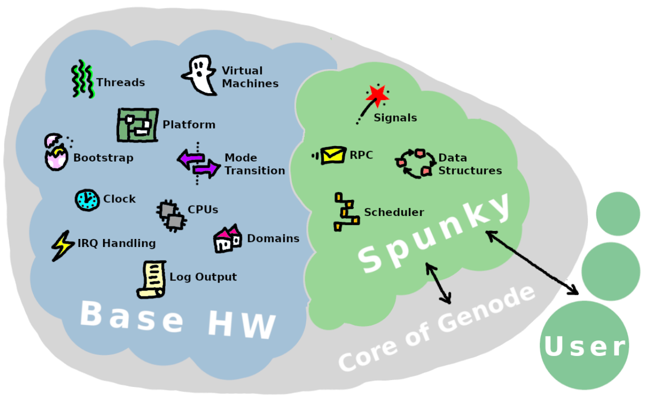


Figure 2: The current state of Spunky

public interface as the C++ class and its purpose is the translation and validation of arguments as required before calling the actual Ada package. This way, the actual Ada package remains pure and, at the same time, integrates seamlessly into the C++ code base.

With this approach at hand, the implementation of Spunky started half a year ago and some of the core aspects of the design are already finished (see Figure 2): The RPC, asynchronous signalling, the CPU scheduler, and the required data structures (two list types). Thanks to the strictness of the language and the GNAT compiler as well as the restrictive compiler flags used for Spunky and its hybrid approach, runtime debugging was not needed at all so far. In fact, each new package integrated could directly deal with complex scenarios without any aftermath. Meanwhile, even the Genode desktop system named Sculpt became available on Spunky [1].

3 Remote Procedure Calls

The RPC state machine of Spunky is a pretty free-standing package, depending only on a generic queue implementation. As RPC endpoints are always threads, the RPC state machine is integrated as record field of the thread kernel-object. I.e., the kernel memory it requires gets completely accounted with the creation of a thread to the thread owner. For each RPC subject or thread the state machine requires merely three states: "Inactive", "Waiting for reply", and "Waiting for request".

If a subject tells the state machine (via a syscall) that it would like to process incoming RPC requests but there are none in the queue, it blocks in "Waiting for request". As soon as there are requests in the queue, the subject is unblocked - becoming "Inactive" again - to process the oldest request in the queue. If a subject has sent an RPC request, it blocks in "Waiting for reply". On nested RPC, the subject blocks in "Waiting for reply" while processing a request from the queue.

Additionally to this, the RPC state machine supports helping. Helping means that an RPC caller lends its scheduling context to the callee for the time of the RPC to speed up the processing of his request. Helping works also nested: If thread A lends its scheduling context to thread B and thread B does an RPC with helping to thread C, thread C temporarily receives the scheduling contexts of both thread A and thread B.

Now, the RPC state machine doesn't know about scheduling contexts. But it holds a flag for each subject to remember whether the subject is helping it's callee or not. When in the above example with threads A, B, and C the scheduling tells Spunky that the scheduling context of thread A is next, Spunky simply asks the RPC who is the final benificiary of thread A's helping. The RPC state machine traverses along the RPC paths with the helping flag set and ends up at thread C. So, thread C would be executed.

4 Asynchronous signalling

Besides the synchronous RPC mechanism, Genode provides fire-and-forget signals as a second method of inter-process communication. In contrast to RPC requests, signals carry no payload data besides a local identifier. Furthermore, in signalling there are two different roles, the signal receiver and the signal context. A signal context represents a specific type of signal, for instance, a timeout that triggers or input data that arrived. At the other hand, a signal receiver refers to the context in which signals are handled, like a network stack handling a protocol timeout or a block device driver handling the reception of requested blocks.

Signal contexts must be registered at the receiver that shall handle them. A receiver may thereby handle multiple signal contexts at a time but a signal context, in turn, can have only one receiver. Finally, when a thread gets notified by a signal receiver that one of its signal contexts triggered, this is called a signal.

In Spunky, both signal context and receiver are free-standing kernel objects that are created, managed and destroyed via dedicated syscalls. They are described together in one package that, again, depends only on a generic queue implementation.

Using this package, a thread can subscribe to a receiver for the reception of a signal. The thread gets then blocked and added to the handler queue of the receiver. At the other side, when someone triggers a signal context, the context gets added to the delivery queue of the corresponding receiver.

Each time a receiver finds both handler and delivery queue non-empty, it selects any thread from the handler queue and sends him a signal for one of the signal contexts from the delivery queue. After this, the thread gets unblocked and removed from the handler queue and the signal context is removed from the delivery queue too. This process is repeated until one of the two queues is empty.

Besides the basic signaling mechanism, Spunky helps the userland to manage the lifetime of signalling objects. A signal context should never be destroyed while getting handled by a thread to avoid dangling references in the userland. For the kernel object of a signal context it is an easy task to keep track of that state. Therefore, if a thread wants to destroy a signal context, it first tells the kernel object to dissolve itself. If the signal context is being handled at this moment, the thread will be blocked and remembered by the context.

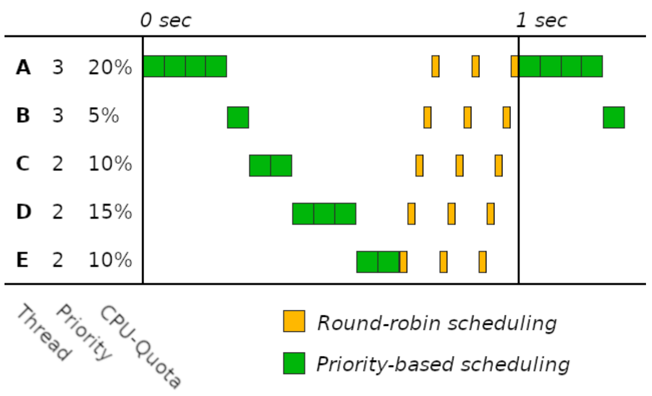


Figure 3: Scheduling scheme for one core in Spunky assuming five threads that wont block

Figure 4: The Sculpt desktop OS on top of Spunky

Furthermore, the signal context will now refrain from entering the delivery queue again. Once the handler of the signal context acknowledged that he is done, the other thread is unblocked and can then safely continue destroying the signal context.

5 CPU scheduling

A Genode thread has three parameters that are of interest for scheduling. Its affinity defines on which cores it may be executed. Its priority defines which of the other active threads it displaces in scheduling and by which of the other active threads it gets displaced. A threads CPU quota finally limits the effect of its priority to a certain percentage of the available CPU time.

For example, if a thread holds a CPU quota of 1 percent, this means that out of, let's say, 1 second of CPU time its priority is active only for 10 milliseconds. The rest of the time the thread gets scheduled en par with all other threads whose priority turned inactive. CPU quota can avoid starvation in a system when a high priority task becomes malicious.

Furthermore, CPU quota can be passed from one Genode component to another. A parent process, for instance will normally pass some of its own CPU quota to its child processes to support their execution.

Not all kernels supported by Genode apply all three of these concepts. Especially the CPU quota comes into effect only on Base-HW and Spunky. When it comes to affinity, however, these two kernels have limited support so far in the way that the affinity of a thread, once defined, remains fixed and that a thread cannot be affine to multiple cores.

In Spunky, for each CPU core there is a dedicated instance of the CPU scheduler and they don't interact with each other. Therefore, the implementation of the CPU scheduler doesn't have to deal with affinities. As with RPC and signalling, the CPU scheduler is implemented as a pretty free-standing package depending only on the generic implementation of a double-linked list.

The scheduler manages scheduling contexts. Each thread has its own scheduling context in form of a record field. But, as already mentioned in the RPC section, it is perfectly fine and desirable that a thread runs on a scheduling context that is not its own. The ownership relation is only meaningful for doing the memory accounting, determining the lifetime, and knowing the default user of the scheduling context. Moreover, a scheduling context can either be ready or unready. A context is ready as long as the current user of the context is not blocked.

Now, if, during a kernel pass, Spunky finds that scheduling is necessary - for instance because a scheduling timeout triggered or because the active thread became blocked - it first tells the scheduler how much time the active thread spent. Then, Spunky asks the scheduler which of the ready scheduling contexts should be next and for how long (the next scheduling timeout). Finally, the current user of the new context is determined (see RPC helping) and executed in the userland.

Under the hood, the CPU scheduler manages scheduling contexts in multiple lists. Each of these lists implements a simple round-robin scheduling amongst the listed contexts. The first list references all ready contexts whose priority isn't active anymore because they have no CPU quota left. Besides that, there is one additional list for each priority level. A list of a priority level references all ready scheduling contexts with this priority. When being asked for the next ready context, the scheduler therefore only has to iterate over these lists beginning from the highest priority level down to the lowest followed by the priority-less list and, on the way, pick the head of the first list that isn't empty.

But CPU quota is a relative value, right? Well, the CPU scheduler sets CPU quota in relation to a super period. This super period currently amounts to 1 second. So, inside Spunky, the CPU quota of each scheduling context is given as absolute value (for instance, 1% would be 10 milliseconds) and at the end of a super period the CPU quota of all scheduling contexts gets refilled.

6 Desktop system based on Spunky

A good scenario for demonstrating the full potential of Spunky is the Sculpt desktop OS that is based on Genode. Sculpt is maintained by Genode Labs as part of the mainline Genode repository since 2018. The default setup starts several dynamic subsystems for featuring sand-boxed device drivers, a graphical control interface called "Leitzentrale", and a so-called "Runtime" for user processes.

On top of this default setup, the user can dynamically deploy further subsystems inside the runtime by using the Leitzentrale for the instrumentation of the Genode package management. Individual runtime setups can easily be stored on a persistant device to be loaded automatically on next startup. There is a broad variety of packages available for the runtime. For instance, a VirtualBox port for using other operating systems safely inside Sculpt, a native window manager, a Qt text editor, the Arora web browser, a graphical and enhanced 'top', a Unix runtime environment, right up to interactive games.

On Spunky this general-purpose system runs fluently. However, note that one limitation of Spunky in this context is that virtualization is not supported so far. Spunky also is not yet part of the official Genode repository. That said, there is no official ready-to-use image maintained that features Sculpt on top of Spunky. However, building such an image from source isn't complex and aligns with the way of how Genode images are built in general. There is an online tutorial on how to build your own Spunky Sculpt that is referenced in section 8.

7 Roadmap

While initially being merely a hobby project of mine, last month, the Spunky project entered the official Genode roadmap [GenodeRoadmap]. The goal till autumn 2020 is to have a kernel completely written in Ada and without any remaining dependencies from the Base-HW kernel. At that point, the design of Spunky is free and might be, in a next step, modified in order to reach SPARK compliance.

Although Spunky was written right from the beginning with SPARK compliance in mind, there are some fundamental obstacles that could not be eliminated while still depending on Base-HW code. The main issue here is the Base-HW memory management that requires the use of access types in Ada so far. Should the re-design succeed, the kernel might become ready for fundamental proofs with SPARK in 2021.

8 Further resources

All the basics about Genode and its generic microkernel interface described in this article can be read in more detail in the Genode Foundations book that is available as free PDF document and as web page:

https://genode.org/documentation/genode-foundations/index

There is a series of articles on the Genode community blogspace that accompanies the development of Spunky in detail:

- "Startup and RPC"
  http://genodians.org/m-stein/2019-05-09-spunky-1
- "Signals and repository integration"
  http://genodians.org/m-stein/2019-12-20-spunky-2
- "CPU scheduler and Sculpt tutorial"
  http://genodians.org/m-stein/2020-02-03-spunky-3

The Spunky source code is currently available on a dedicated topic branch on GitHub:
https://github.com/m-stein/genode/commits/3308_spunky_a_kernel_using_ada

The presentation about Spunky during the FOSDEM 2020 was run completely on the mentioned Sculpt desktop system with Spunky as kernel. The recording of this presentation is available online:

http://bofh.nikhef.nl/events/FOSDEM/2020/AW1.125/ada_spunky.mp4

Furthermore, there is a dedicated GitHub issue that provides space for discussions around Spunky:

https://github.com/genodelabs/genode/issues/3308

References

[1] Martin Stein, A tutorial for building the Sculpt desktop system using Spunky, *http://genodians.org/m-stein/2020-02-03-spunky-3*.

[2] Genode Labs, Official webpage of Genode Labs, *https://genode-labs.com/*.

[3] Martin Stein, Recording of the presentation of Spunky at FOSDEM 2020, *https://fosdem.org/2020/schedule/event/ada_spunky/*

[4] Genode Labs, Documentation of the Sculpt desktop system, *https://genode.org/download/sculpt*

[5] Alexander Tarasikov, Porting Genode to commercial hardware, *https://allsoftwaresucks.blogspot.com/2013/05/porting-genode-to-commercial-hardware.html*

Integrated Formal Analysis for Ada Programs

Tewodros A. Beyene, Christian Herrera
fortiss - Research Institute of the Free State of Bavaria, Guerickestraße 25, 80805 München, Germany;
email: beyene@fortiss.org, herrera@fortiss.org

Abstract

Formal software analysis tools are playing an essential role in software developers' efforts for removing errors during development. In this short paper, we present a two-step approach for Ada code review, where light-weight Ada code analysis is combined with more involved formal analysis using an Ada model checker. We have automated this process using an existing tool integration framework, and our experimental evaluation shows that such approach is able to significantly reduce false positives from the analysis result and thereby improving the quality of the code review.

Keywords: Code Review, Static Analysis, Model Checking.

1 Introduction

Code review is an important activity in software development where source code is systematically examined with the main goal of finding errors overlooked during development, and ensuring the code is well-structured and maintainable. A review process may also try to ensure compliance of the code with language and organization specific coding guidelines. The ultimate goal of code review is to improve the overall quality of the software being developed.

There exist various methods for conducting code reviews, ranging from the highly structured formal code reviews to informal and collaborative light-weight code reviews. Formal code review methods, e.g., Fagan inspection [1,2], are traditional methods based on line-by-line inspection of the code in a series of meetings by a team of experienced developers and dedicated reviewers. As programs grow, however, these methods become lengthy and involved. Light-weight code review methods, e.g., over-the-shoulder review, email pass-around and pair programming [3], typically require less overhead than formal code review, and they can be equally effective when done properly [4]. Tool-assisted code review, which involves using software tools, can also be considered as light-weight code review. Tool-assisted methods often automate parts of the review process such as review scheduling, source code sharing, metrics collection and quick communication among reviewers [3]. Although such automation helps in making the review process effective and efficient, the code reviewing task itself and review report generation is mostly done manually by code reviewers.

Industrial Presentation submitted for the Ada-Europe International Conference (AEiC 2020).

There have been recent efforts to further automate code review process by leveraging formal software analysis tools. One example is SCRUB [5], a tool that combines peer code review with tool-generated reviews which is developed at NASA JPL [6] for supporting an effective tool-based code review process. The ability of the tools to analyse large source code in shorter time, as compared to a human reviewer, makes such tools successful. However, as most static analysis tools may return false positives, review process based on static analysis may check by mistake some part of the code as erroneous.

In this paper, we present a two-step approach for Ada code review, where light-weight Ada code analysis is combined with more involved formal analysis using an Ada model checker. Many certification standards not only include review as one of the recommended verification techniques but also encourage the use of combination tools in verification workflows. For example, DO-178C gives special attention to the use of analysis tools in combination to achieve better analysis results and higher confidence in the results [7]. *"The use of tools or combinations of tools and parts of the software development environment should be chosen to achieve the necessary level of confidence that an error introduced by one part would be detected by another"* ([7] Sec. 4.4.1 (b)). Our approach stems out of these recommendations to enable better Ada code review method with higher confidence on the correctness of the final review report.

The rest of this paper is structured as follows. In Sec. 2, we describe our code review approach followed by its illustration using a small example in Sec. 3. The approach is evaluated on a set of benchmark Ada programs in Sec. 4. The paper finally presents related works in Sec. 5.

2 Approach

Our two-step code review approach is given in Figure 1. In the first step, the GNATProve static analyser is employed to check runtime errors such as *division by zero*, *integer overflow/underflow*, *floating-point overflow/underflow*, *array out of bound errors*, etc. This step generates an initial review report containing potential errors produced by GNATProve. Error descriptions in the report consist of values for the following fields: $error\_type$, $file$, $line\_number$, $error\_value$, etc. In the second step, the code review process employs more powerful formal analysis using the AdaHorn model checker [8], by using error descriptions from the initial review in order to generate a more refined report. That is, false positives that were included in the initial report will be removed from the final report. In addition, potential errors from the initial report that are found to be real errors by the refinement step will have a more detailed error description. Note that model checking the code directly will be very expensive. Therefore, we are using model checking only to refine the results of static analysis, which is more light-weight but less precise than model checking.

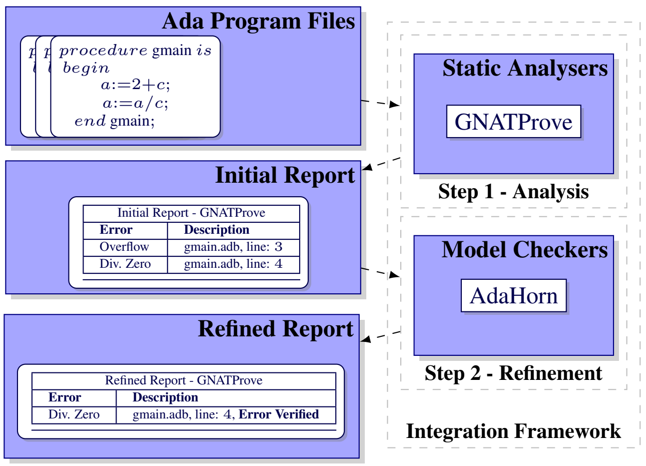


Figure 1: A two-step code review approach

As the approach uses the ETB framework [9, 10] to integrate Ada analysis tools, wrappers needs to be written for each analysis tool. The role of a wrapper is to translate inputs from the framework to the input language of the tool and outputs of the tool to a standard structure expected by the integration framework. The framework can also be used to integrate tools that support checks for Ada coding guidelines, e.g., AdaControl. However, as such checks are not amenable to refinement, we have not included them in this work.

3 Illustration

In this section, we illustrate our code review approach by using a small example, which is shown in Figure 2. We specifically show how results from a code analysis can be refined by applying a model checker to keep only real issues in the final code review report.

Our example Ada program consists of the following three files:

- a main file (Figure 2a) for calling procedure *Quarter* (line 8).
- a specification file for Quarter (Figure 2b). This file initializes the variable *months_qtr* to 3, i.e., the number of months in any quarter, (line 2).
- an implementation file for Quarter (Figure 2c). This file initializes to 3 the variable *qtr*, which stores a given quarter of the year (line 3), and to 0 the variable *months_in_qtrs* (line 4). Line 6 stores in months_in_qtrs the number of months occurring from the first until the quarter stored in qtr. Line 7 shows a division.

Assume we would like to review if two runtime properties for the implementation file (Figure 2c). The two properties are absence of arithmetic overflow in line 6, and absence of division by zero in line 7.

```
1  with Qter_of_year;
2  use Qter_of_year;
3  procedure gmain is
4  begin
5     Qter_of_year.Quarter;
6  end gmain;
```

(a) gmain.adb

```
1  package Qter_of_year is
2    months_qtr:  Integer :=3;
3    procedure Quarter;
4  end Qter_of_year;
```

(b) qter_of_year.ads

```
1  package body Qter_of_year is
2    procedure Quarter  is
3       qtr:  Integer :=3;
4       months_in_qtrs:  Integer :=0;
5       begin
6          months_in_qtrs:=  qtr*months_qtr;
7          months_in_qtrs:=  12/( qtr  months_qtr);
8    end Quarter;
9  end Qter_of_year;
```

(c) qter_of_year.adb

Figure 2: An example Ada program

After executing the first step of our proposed approach, we get an initial report containing a potential error for each of the above properties. For instance, the following error description is given for the potential arithmetic overflow error in line 6:

```
{"file": "qter_of_year.adb",
 "entity": "Qter_of_year.Quarter",
 "line": 6,
 "rule": "VC_OVERFLOW_CHECK",
 "cntexmp": [
 {"kind": "variable", "name": "qtr", "value": "3"},
 {"kind": "variable", "name": "months_qtr", "value":
        "-1073741825"}]}
```

From this description, we obtain the error location (in fields $file$, $entity$ and $line$), error type (in field $rule$), and counter example information (in field $cntexmp$). The counter example information states that the multiplication in line 6 of the file qter_of_year.adb yields an arithmetic overflow whenever the values for the variables qtr and $months\_qtr$ are 3 and -1073741825, respectively.

In the second step, the error description will be used to encode an input specification for the AdaHorn model checker and the model checker reevaluates if the potential error from the initial report is indeed an error. This step proves that the error does not actually exist in the program, and concludes that the error is a spurious one, i.e., false positive. That is, in file qter_of_year.ads months_qtr is set to 3, and nowhere else updated, thus, this variable is never set to -1073741825 as GNATProve indicates. However, AdaHorn proves that a division by zero error in line 7 actually exists, and concludes that the denominator resulting from the subtraction is 0. Thus, the potential error in the initial review report will also be part of the refined review report.

4 Experimental results

We have evaluated the proposed code review approach using benchmark programs from Beyene et al's work [8] on model checking Ada programs. These programs are grouped into 3 categories, i.e., *arrays*, *floats*, and *loops*, depending on certain data types and programming constructs used in the programs. Our experiment is aimed at capturing the following four common types of runtime errors in Ada programs: *division-by-zero errors*, *integer overflow/underflow*, *floating-point overflow/underflow*, and *array-out-of-bound errors*. The proposed approach is evaluated using 20 programs from each category, making the total number of programs in our experiment 60. Programs in each category are also classified into *error-free* and *erroneous*. A program is said to be *erroneous* if it has one of the runtime errors described above. Otherwise, it is classified as *error-free*. For example, from the 20 programs in the *loops* category, 13 are *error-free* and 7 are *erroneous*.

The code review approach first applies static analysis using GNATProve. This tool checks each program for a potential runtime error and provides the verdict *"proved"*, when it verifies there is no any error in the program. If the tool suspects a potential error, it provides the verdict *"might fail"* with a description of the potential error. These potential errors appear in the initial review report of the Ada programs. The initial review report for the benchmark programs, including the number of errors for each program category, is given in Table 1. For example, from the 13 error-free programs in the *loops* category, GNATProve returns *"proved"* for 2 programs and *"might fail"* for the remaining 11 programs, i.e., GNATProve returns 11 false positives for this program category. For the corresponding 7 *erroneous* programs, GNATProve returns *"might fail"* for each one, which is the right verdict. GNATProve returns 52 of the total 60 programs as *"might fail"*. Note that there are only 19 *erroneous* programs, i.e., a total of 33 false positives.

In the second step, also called a refinement step, the AdaHorn model checker is used to refine the initial review report, by re-evaluating if each of the error in the report is indeed a real error or a false positive. AdaHorn returns *"sat"* if the potential error does not actually exist in a given program, and returns *"unsat"* if the potential error indeed exists in the program. The model checker can also return *"unknown"*, when it cannot conclude sat or unsat. Encoding each error from the initial review report in the input language of AdaHorn will be done by a wrapper function added to the integration framework. The refinement step is able to identify that from the 52 potential errors in the initial error report, 25 of them are false positives. Therefore, all these false positives will be removed from the final error report that will be shown to developers.

The refined review report for our benchmark programs is given in Table 2. Note that the refinement step is also able to correctly identify *erroneous* programs from all of the three categories as *"unsat"*. For instance, while the refinement step was very efficient in identifying all 11 *error-free* programs as *"sat"* for the *loops* category, it was not so efficient for the *floats* category as it is able to identify only 4 of the 10 *error-free* programs as *"sat"*.

For this experimental evaluation, our code review process takes less than 4 seconds for generating the initial review report, and around 5 minutes for the final report, which including a 60 seconds timeout for some examples by AdaHorn. Our benchmark programs have at most 20 lines of code. One direction of future work is to evaluate scalability of the approach for larger programs.

5 Related work

In this section, we discuss selected approaches for Java, C and C++ that share similarities with ours. That is, they combine light-weight and more involved techniques for reducing false positives in analysed programs.

The work in [11] checks for errors in Java programs, by using in a first step the static analyser ESC/Java for detecting errors, for instance, division by zero or accessing an array out-of-bounds. That tool produces error reports that detail abstract conditions (constraints on program values) necessary for errors to occur. Given that ESC/Java may produce spurious error reports by inaccurately modeling the Java semantics, in a second step, constraint solvers are used to derive concrete input values that satisfy those abstract conditions. In a third step, the tool JCrasher is used to generate test cases using the previously derived concrete values. If test cases do not result into a runtime exception, then the concrete value is considered as a false positive.

The work in [12] checks for errors in C and C++, by using in a first step *data flow analysis* to restrict (in a reasonable amount of time and memory) an extensive original source code to fractions that are relevant to find a given error. The inherent approximate nature of this kind of analysis leads to false positives. As a second step theorem provers are used to check feasibility of found errors. The authors of this work implement those steps in the Orion tool, and obtain a fast and fully automatic analysis, however, they recognize that their tool is unsound by not ruling out each false positive found in the first step.

The work in [13] uses the industrial static analyser Polyspace in a light-weight analysis of errors in C and C++ programs. Given that Polyspace suffers from a high rate of false positives, the authors in that work use the bounded model checker CBMC to refine those false positives in a next step. However, before using CBMC programs must be manually annotated, for instance, by providing invariants or input constraints. This yields a semi-automatic analysis that is still able to refine a significant number of false positives. Note that our approach uses as well a static analyser for a light-weight analysis, and a bounded model checker for refining output false positives. However, no user interaction is required for that refinement.

References

[1] M. E. Fagan, "*Advances in Software Inspections*," *IEEE Trans. Softw. Eng.*, 1986.

[2] M. E. Fagan, "*Design and Code Inspections to Reduce Errors in Program Development*," *IBM Syst. J.*, 1999.

[3] J. Cohen, "*Best Kept Secrets of Peer Code Review (Modern Approach. Practical Advice.)*," 2006.

[4] J. Cohen, "*11 Proven Practices for More Effective, Efficient Peer Code Review*," 2011.

[5] G. J. Holzmann, "*SCRUB: A Tool for Code Reviews*," *Innov. Syst. Softw. Eng.*, 2010.

programs category	program type	number of programs	GNATProve analysis verdict		potential errors
			"proved"	*"might fail"*	
arrays	error-free	12	0	12	20
	erroneous	8	0	8	
	total	20	0	20	
floats	error-free	14	4	10	14
	erroneous	6	2	4	
	total	20	6	14	
loops	error-free	13	2	11	18
	erroneous	7	0	7	
	total	20	2	18	

Table 1: Initial code review report

programs category	program type	number of programs	AdaHorn refinement analysis verdict			number of issues
			"sat"	*"unsat"*	*"unknown"*	
arrays	error-free	12	10	1	1	10 issues
	erroneous	8	0	8	0	
	total	20	10	9	1	
floats	error-free	10	4	3	3	10 issues
	erroneous	4	0	3	1	
	total	14	4	6	4	
loops	error-free	11	11	0	0	7 issues
	erroneous	7	0	7	0	
	total	18	11	7	0	

Table 2: Refined code review report

[6] K. Havelund and G. J. Holzmann, "*Software Certification: Coding, Code, and Coders*," EMSOFT, 2011.

[7] "RTCA DO-178C Software Considerations in Airborne Systems and Equipment Certification. RTCA Standard," December 2011.

[8] T. A. Beyene, C. Herrera, and V. Nigam, "*Verification of Ada Programs with AdaHorn*," *Ada User Journal, Volume 40*, 2019.

[9] T. A. Beyene and H. Ruess, "*Evidential and Continuous Integration of Software Verification Tools*," in *FM 18*, 2018.

[10] S. Cruanes, G. Hamon, S. Owre, and N. Shankar, "*Tool Integration with the Evidential Tool Bus*," in *VMCAI*, 2013.

[11] C. Csallner and Y. Smaragdakis, "*Check 'n' Crash: Combining Static Checking and Testing*," in *ICSE 05*, ACM, 2005.

[12] D. Dams and K. S. Namjoshi, "*Orion: High-Precision Methods for Static Error Analysis of C and C++ Programs*," in *FMCO 05*, 2005.

[13] H. Post, C. Sinz, A. Kaiser, and T. Gorges, "*Reducing False Positives by Combining Abstract Interpretation and Bounded Model Checking*," in *ASE 08*, 2008.

De-RISC – Dependable Real-Time Infrastructure for Safety-Critical Computer Systems

Francisco Gómez, Miguel Masmano, Vicente Nicolau
fentISS, Ciudad Politécnica de la Innovación, building 9B office 0.74, Camino de Vera s/n, 46022 Valencia, Spain

Jan Andersson
Cobham Gaisler, Kungsgatan 12, SE-411 19, Göteborg, Sweden

Jimmy Le Rhun
Thales Research and Technology, 1 Rue Augustin Fresnel, 91120 Palaiseau, France

David Trilla, Felipe Gallego, Guillem Cabo, Jaume Abella
Barcelona Supercomputing Center (BSC), c/ Jordi Girona, 31, 08034 Barcelona, Spain

Abstract

The space domain demands increased performance, reliable and easy to verify and validate platforms to match the requirements of highly autonomous missions and systems that need to undergo qualification and certification against safety guidelines, and be commercialized worldwide minimizing export restrictions. Unfortunately, commercial platforms either fail to match domain-specific requirements for space (e.g. safety requirements), are limited by US export regulations, or simply fail both sets of requirements.

This paper introduces De-RISC, a novel HW/SW platform meeting space requirements for safety- and mission-critical applications by construction, with explicit support to ease performance validation and diagnosis, and based on the RISC-V instruction set architecture. The De-RISC platform, which builds upon fentISS' XtratuM hypervisor and a Cobham Gaisler (CG) NOEL-V based MPSoC, will reach commercial maturity in 2022, and will be assessed against a space use case.

Keywords: Space, RISC-V, safety, hypervisor, MPSoC

1 Introduction

Space market requirements for safety- and mission-critical applications increase in several directions: (1) higher performance is needed to support increasing autonomy of spacecraft; (2) safety features need to hold in the context of multicores and mixed criticality; (3) high reliability must be achieved without jeopardizing performance; (4) performance validation and runtime diagnosis must be attainable in multicore SoCs; and (5) export restrictions must be minimized to enable commercialization and wide adoption of the technology.

Most of those requirements – namely 2, 3 and 4 – relate to the adoption of multicore processors in the space domain to reach the desired performance levels (requirement 1), whereas some others – namely 2 and 5 – relate to commercial reasons such as enabling the integration of software from different providers safely and allowing adoption and integration by avoiding proprietary IP and closed standards.

Existing technology fails to achieve some of those requirements. In particular, as detailed later, existing commercial platforms, including hypervisors and systems on chip (SoCs), do not match some of the aforementioned requirements: they are often single-core platforms, fail to match space requirements related to safety and/or reliability, provide limited support for performance validation and runtime diagnosis, and build upon proprietary Instruction Set Architectures (ISAs), such as x86 and Arm, which pose export restrictions. In this context, a need for a new platform meeting the 5 key requirements described before emerges.

This paper presents De-RISC, a new HW/SW platform for the space domain meeting those 5 requirements by construction. The cornerstone of De-RISC is the combination of powerful commercial products for the space domain (XtratuM hypervisor from fentISS and NOEL-V MPSoC from Cobham Gaisler) with an increasingly popular and non-proprietary ISA (RISC-V). Such brand new platform, which inherits technology developed and validated for SPARC during decades, is further enhanced with explicit hardware support for time predictability and performance validation building on the developments and experience provided by a key space and avionics end user (Thales) and a technology provider with a long track of experience in the space domain (Barcelona Supercomputing Center, BSC).

Industrial Presentation submitted for the Ada-Europe International Conference (AEiC 2020).

The De-RISC platform, which stands for **De**pendable **R**eal-time **I**nfrastructure for **S**afety-critical **C**omputer Systems, is expected to be fully developed and integrated by early 2021, go through extensive validation at functional, reliability and performance levels during 2021, and be ready for commercialization in early 2022 building upon appropriate FPGAs. The ASIC version is expected to arrive soon after that.

The rest of this paper is organized as follows. Section 2 provides some state of the art of existing space technology and RISC-V developments. Section 3 introduces the XtratuM hypervisor part of De-RISC. Section 4 introduces the NOEL-V based multicore SoC part of De-RISC. Section 5 describes the extended hardware support for time predictability and performance validation. Section 6 presents the use case which will be used to perform the final platform validation. Finally, Section 7 concludes this work.

2 State of the art

In this section we review the state of the art on hypervisors for the space domain, space-grade microprocessors, and RISC-V based processors.

2.1 Space-grade hypervisors

Although hypervisors are used in some conventional Space missions, 'NewSpace' is bringing new opportunities to hypervisors in the space domain. Constraints in size, weight, cost, and power consumption are not so stringent in conventional space missions. A conventional satellite can easily weight a few tonnes and require enough budget to be built up from scratch. In comparison, typical satellites of NewSpace weight less than 200 Kg, but they are deployed in large quantities of tens to even thousands of satellites per mission [15].

European hypervisors for space. (1) XtratuM, provided by fentISS, is the hypervisor with the highest presence in the space domain at European level. It is already orbiting in seventy six satellites of three different missions and it is already selected for use in several conventional missions and new space missions over the next five years. XtratuM is designed to guarantee temporal and spatial isolation of safety-critical systems, thus enabling the development of cost-effective solutions requiring mixed-criticality systems. (2) Air, a hypervisor by GMV-Portugal, has been tested in 2 projects to demonstrate feasibility and performance in monocore and multicore platforms [9] and it is foreseen to flight in the framework of the INFANTE project. Other products implementing time and space partitioning, such as (3) PikeOS by Sysgo [2], made some steps to be present in space missions.

Worldwide hypervisors for space. (4) LynxSecure by Lynx Software Technologies (US) is a separation kernel hypervisor designed for safety and security critical applications. (5) Wind River Hypervisor (US) is a type 1 embedded hypervisor with a very small footprint, minimal latency, deterministic capabilities, and optimizations for maximum performance. Its safe and secure partitioning capability is designed to isolate and separate applications of mixed levels of criticality and decouple the life cycle of certified and non-certified applications to mitigate recertification costs in safety-critical systems. Despite of not being a hypervisor, Green Hills (US) has (6) INTEGRITY, a RTOS designed to meet requirements for security, reliability, and performance.

2.2 Space-grade microprocessors

State-of-the-art Commercial Off-The-Shelf (COTS) devices can be included in experimental LEO missions and mega-constellations, where availability of a single spacecraft is less critical than for a science space mission using only one spacecraft. These devices are, however, not suitable for most deep space missions, as opposed to De-RISC architecture, which will be tailored for space-grade FPGAs in a first iteration, and to a hardened ASIC in a second stage.

European devices. There exist several European space-qualified microprocessors today, the majority of them SPARC-based: (1) ATMEL ERC32 was the first European 32-bit space microprocessor. (2) ATMEL AT697F consists of a LEON2FT processor with UART, PCI and simple GPIO interfaces. However, avionics systems typically require additional interfaces (e.g. SpaceWire and MIL-STD-1553B). (3) Cobham Gaisler's GR712RC is a dual-core LEON3FT processor [4] that provides additional interfaces used in spacecraft avionics compared to the AT697F. (4) Cobham Gaisler's GR740 is the most recent and most powerful LEON4FT processor [5], and provides further external interfaces for larger and higher bandwidth memory using existing space-grade memory devices.

Worldwide devices. BAE Systems (US) offers the RAD750 and RAD5545 devices, and DDC/Maxwell (US) the SCS750 board. They are based on the PowerPC ISA, a proprietary ISA. One particular mission that was forced to use US components due to lack of a European alternative was GAIA (Global Astrometric Interferometer for Astrophysics, ESA mission for space observation). Studies have now shown that the GR740 provides enough performance to replace the PowerPC processors in the GAIA application [6]. The De-RISC hardware platform builds on technology and lessons learned from the GR740 and delivers even higher performance.

Currently, there are several microprocessors under development by the NASA, Boeing Corporation, H2020 DAHLIA, and Ramon Chips. Those developments are at different maturity levels, and thus their final performance is unknown. However, a commonality for all of them is that they build upon proprietary ISAs (e.g. Arm).

2.3 RISC-V cores and microprocessors

The openness of RISC-V has led to a number of institutions developing cores and SoCs based on RISC-V. Only in the RISC-V International website there are already 56 cores and 30 SoCs announced [12]. SiFive, UC Berkeley, Andes, CloudBEAR, Codasip, Microchip, and ETH Zurich/U. Bologna are among the most active suppliers, with SoCs and cores such as those based on Rocket (SiFive and UC

Berkeley), E76-MC (SiFive), and Ariane and PULPino (ETH Zurich and U. Bologna) being highly popular due to the already long record of activity of their suppliers in the field of RISC-V designs.

However, a common trade across all those designs is that they are not specifically devised for critical real-time systems, which impose strong requirements in terms of functional safety and robustness. In particular, SiFive's E76-MC is the embedded processor closest to the needs of safety-critical systems, but still misses many features such as support for domain-specific interfaces, watchdogs, and the like [13].

Also, none of them targets the space domain, which poses additional reliability requirements due to the high radiation levels and the impossibility to perform any recall in space. Therefore, those designs would need deep modifications to make them amenable for space applications.

3 RISC-V compliant XtratuM hypervisor

On the software side, as part of De-RISC, the XtratuM Next Generation (XNG) hypervisor [8] will be ported to the hardware RISC-V architecture and the required run-time systems, in particular the ARINC-653 compatible LithOS run-time by fentISS, will be ported on top of the RISC-V XtratuM. The hypervisor will be extended to support the multicore configuration of the De-RISC hardware platform avoiding time interference when using shared resources by means of the mechanisms provided by the hardware.

XNG is a complete rewrite of the XtratuM hypervisor that builds on the accumulated expertise after working for more than 12 years in the development of the XtratuM variants. Its building blocks provide the services needed by safety-critical systems such as partition management, support for system and non-system partitions, resource virtualization, temporal partitioning based on a cyclic scheduling policy, spatial partitioning based on the support provided by the hardware, inter-partition communication through sampling and queuing ARINC-653 like ports, the XtratuM configuration (XCF, a set of XML files), the health monitoring service (which detects faults in the hardware and in XNG itself) and observability of the system.

Figure 1 shows a typical configuration with XtratuM running on top of the hardware platform developed in De-RISC: the block diagram illustrates a XtratuM-based system architecture with a first partition (1) containing a bare XtratuM Run-time Environment (XRE) with an application and two partitions (2 & 3) containing their corresponding OSs and their corresponding applications. XRE is a C library component provided with XtratuM for simplifying the interaction of the application code with the low-level management required by the Partition Virtual Execution Environment (PVEE).

LithOS is a para-virtualised guest operating system which uses the services provided by XtratuM to offer an ARINC-653 APEX to the applications. While XtratuM provides the partitioning and inter-partition communication mechanisms, LithOS builds on top of them to provide a complete run-time solution by including processes, inter-process communication, synchronization mechanisms, error handling and the rest of functions expected from an ARINC-653 compliant run-time.

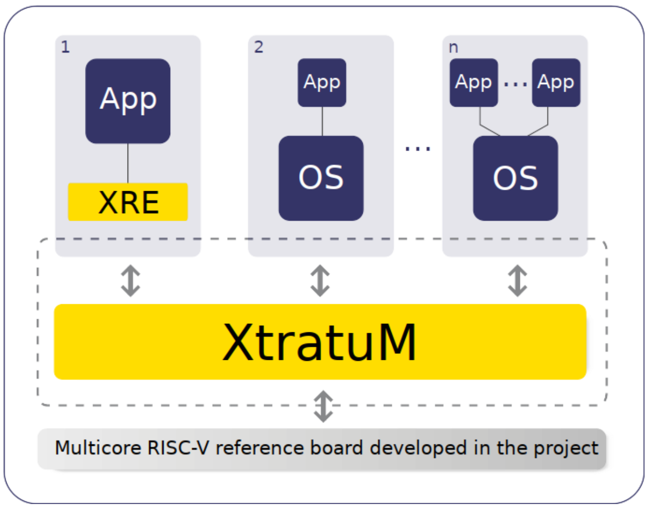


Figure 1. XtratuM running on top of the De-RISC hardware platform

Development and debugging tools will also be available as in any XtratuM distribution. This comprises Xoncrete Scheduling Tool (to analyse system schedulability and produce static feasible schedules), xcparser (to translate a XtratuM XML configuration to binary code), xci (to interpret the content of the hypervisor data memory through the hypervisor observability symbols) and xcon (to observe the memory area of the hypervisor console).

The XNG and LithOS porting will be carried out following the ECSS development process to provide the essential elements for a space qualification at ECSS level B. This will make the hypervisor ready for the aerospace market right after the project completion.

4 RISC-V based NOEL-V multicore SoC

The De-RISC SoC architecture topology is shown in Figure 2 and shows an architecture that has similar traits to what is available in the commercial domain. General-purpose processing capabilities are provided by the GPP elements, each one constituting a multicore processor system in itself.

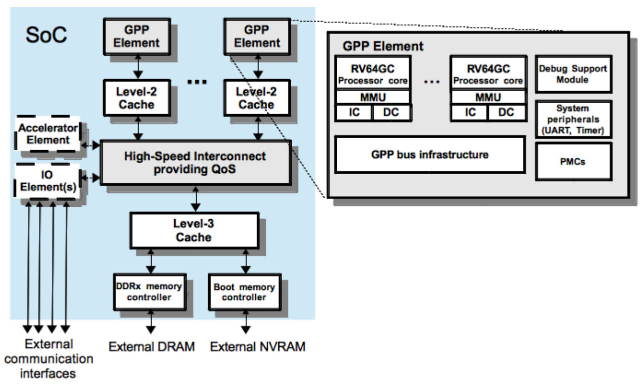


Figure 2. De-RISC MPSoC architecture topology

Depending on target technology and foreseen application needs, a trade-off can be performed between the number of GPP elements and the number of processor cores within each element. The current baseline De-RISC design has four general purpose processor cores per GPP element.

The GPP elements are connected to the larger system through a dedicated Level-2 cache. The Level-2 cache connects to a high-speed interconnect that also connects an IO subsystem, Accelerator subsystem and finally a Level-3 cache that in turn connects a Memory subsystem.

The MPSoC has been designed with fault-tolerance concepts applied in the same way as for the LEON line of processors. Several other RISC-V implementations take a "data center" approach to fault-tolerance where errors in computations are detected and software then rolls back the computation. In some cases the error detection can take thousands of clock cycles, with a resulting timing jitter in software execution and possibility of propagating bad state to other parts of the system. The approach taken in LEON systems is to allow seamless correct operation in the presence of faults. Correct operation is maintained by correcting errors on the fly. In the event of uncorrectable errors, execution is immediately stopped, and corrupted state is never allowed to propagate outside of the component. This approach is taken both for the NOEL-V processor cores and for the on-chip peripherals in the design.

4.1 NOEL-V RV64 processor core

The NOEL-V is a synthesizable VHDL model of a 64-bit processor that implements the RISC-V architecture. The processor is the first released model in Cobham Gaisler's RISC-V line of processors that complement the LEON line of processors.

The NOEL-V is dual-issue, allowing up to two instructions per cycle to be executed in parallel. To support the instruction issue rate of the pipeline, the NOEL-V has advanced branch prediction capabilities. The cache controller of the NOEL-V supports a store buffer FIFO with one cycle per store sustained throughput, and wide AHB data width support to enable fast stores and fast cache refill.

The NOEL-V is interfaced using the AMBA 2.0 AHB bus (subsystem with Level-2 cache and AXI4 backend is also available) and supports the IP core plug&play method provided in the Cobham Gaisler IP library (GRLIB). The processor can be efficiently implemented on FPGA and ASIC technologies and uses standard synchronous memory cells for caches and register file.

The processor model is available as part of the GRLIB IP Library. The GRLIB IP Library is an integrated set of reusable IP cores, designed for SoC development. The IP cores are centered around the common on-chip bus, and use a coherent method for simulation and synthesis. The library is vendor independent, with support for different CAD tools and target technologies. A unique plug&play method is used to configure and connect the IP cores without the need to modify any global resources.

The design flow and portability enabled by the GRLIB IP library allows SoC designs based around the NOEL-V and GRLIB IP to be rapidly targeted towards different target technologies. The initial target for the De-RISC platform are Xilinx Kintex Ultrascale FPGAs. Foreseen future implementations also include ASIC target technologies.

4.2 Communication interfaces and peripherals

The MPSoC platform includes standard peripherals such as interrupt controllers, timer units and system UARTs. The peripheral unit interfaces are implemented according to available specifications to ensure portability between systems and allow for software re-use.

The De-RISC platform supports several communication interfaces. The MPSoC platform is modular and the De-RISC FPGA platform allows the number and type of external interfaces to be adapted to application needs.

The primary memory interface is DDR3 SDRAM complemented by a strong EDAC code that can tolerate loss of functionality of complete external memory components. Boot memory is supported through a parallel memory interface to support existing MRAM and NOR Flash devices. Boot via a SPI memory interface is also supported. The MPSoC also has a NAND Flash interface to provide a large non-volatile memory storage.

For external communication links, the platform supports the interface set provided by the GR740 microprocessor, except for parallel PCI, and extends this with interfaces that are today typically implemented in a companion FPGA next to the GR740. The communication interfaces include:

- High-Speed Serial Link support through SpaceFibre controllers
- SpaceWire communication links, connected to an on-chip router
- 10/100/1000 Mbit Ethernet interfaces
- MIL-STD-1553B
- SPI, I2C, UARTs, GPIO

For the accelerator subsystem, the De-RISC project does not currently define any specific type of accelerators and accelerator cores can instead be added depending on application needs. For future ASIC implementations, general purpose compute engines have been identified to allow for acceleration of a wide range of applications.

5 Extended support for multicores

Given the increasing need to deploy multicores to execute the most critical functionalities in space, and the stringent needs in terms of real-time (guaranteed) performance and validation means in those systems, the De-RISC hardware platform incorporates the following features:

1. Time-predictable and fair access to hardware shared resources across cores.

2. Facilities for performance validation and diagnosis building upon extended interference-aware performance monitoring units (PMUs).

Next, we review the main characteristics of both features.

5.1 Support for time predictability

As shown in recent work [11], resources such as shared cache memories, implementing fair – a priori – features, fail to achieve such fairness due to lack of fairness in some of their interference channels. For instance, space partitioning is usually deployed in multicore SoCs for critical domains, such as, for instance, the NXP T2080 SoC [14], increasingly considered for safety applications in the avionics domain. However, although cache ways can be allocated independently to different cores, request queues for cache accesses, fill buffers, miss queues, etc. are not partitioned, thus leading to large interference. The De-RISC MPSoC will be designed with queues and buffers ensuring that, upon an access to a shared hardware resource, the oldest request from one core will only have to wait for, at most, one request from each other core to be served first in any queue or buffer.

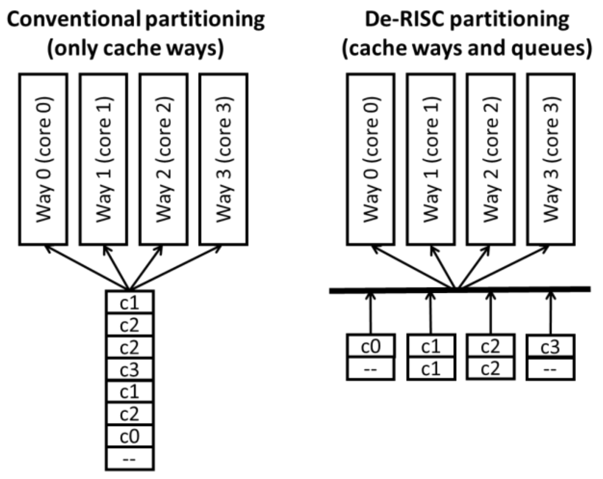


Figure 3. Example of conventional cache partitioning (left) and De-RISC cache partitioning (right)

Support for time predictability is visually illustrated with the example in Figure 3. As shown in the left, conventional partitioning only considers space partitioning for cache storage (i.e. data and tag arrays), but not for queues used to interface the cache. Instead, the De-RISC approach builds upon hardware mechanisms that ensure fair arbitration across requests from different cores. Note that the actual implementation of the mechanism may differ from that in the picture, but its timing behaviour is analogous, i.e. granting one request from each core in a round-robin fashion rather than operating in a FIFO scheme as in conventional designs (e.g. NXP T2080).

5.2 Support for performance validation and diagnosis

PMUs often provide information such as access counts and stall cycles experienced, but they generally lack any information on what master has caused those stall cycles, whether those stall cycles have been spent due to interference in a particular resource, and what maximum latencies individual requests may experience in each resource. This information is of prominent importance for timing verification and validation of critical real-time systems. In particular, maximum latencies allow obtaining empirical evidence of how much a request may be delayed due to interference, as needed to estimate timing bounds for critical tasks. On the other hand, information on where stall cycles are spent, and due to what master, is critically important to gather timing evidence during validation tests on whether estimated bounds have not been exceeded, i.e. per resource and per contender core, thus avoiding the case where non-broken down measurements hide specific overruns that may lead to deadline misses during operation. Also, stall cycles breakdowns allow diagnosing deadline overruns during operation providing precise information on what core and in what resource caused excessive interference, thus easing the process of taking corrective actions to avoid further overruns. Else, corrective actions could be simply naive, or just not being possible due to a failure to identify the cause of the overrun. In De-RISC, we implement those features in line with the proposals in [10], [3].

6 Space use case

In order to validate the complete De-RISC platform from an end-user point of view, several test applications will be ported.

6.1 Basic test benches and compliance tests

The first step to assess the functionality of the processor core is to ensure its compliance with the ISA standard, and to estimate its nominal performance.

For the former, RISC-V International maintains a set of compliance test suites. Running those will allow us to validate the correct implementation of all instructions.

For the latter, core performance benchmarks such as Dhrystone or EEMBC CoreMark, while imperfect as they mostly measure the integer pipeline performance and do not consider parallelism in multicore systems nor exercise the memory hierarchy, remain industry standards. Their results will allow basic performance comparison with other RISC-V implementations and legacy SPARC processors.

6.2 Compute-intensive space application

To assess the performance of the multicore platform and not only one processor core, a more computation-intensive test application is needed. The first goal is to exercise the memory hierarchy with large data, deploy parallel tasks on several cores, and use hypervisor-based or RTOS-based resource management services.

This will also allow a first evaluation of isolation techniques, and the validation of new safety-oriented monitoring resources. In particular, an early estimation of interference effects on interconnect and shared cache could be performed by having several identical memory-intensive tasks mapped on different cores.

The application selected for this validation is a lossless multispectral image compression algorithm, standardized by the Consultative Committee for Space Data Systems as CSSDS-123 [7].

6.3 Representative Space system application set

The final step of validation will involve a set of partitioned tasks representative of a satellite system, interacting through standard inter-partition communication mechanisms.

The Command & Data Handling Platform [1] is inherently multicore, and therefore a good candidate to assess timing interference, the time isolation property and safety hardware components of the De-RISC platform ensuring real-time requirements from the space standard in a multicore context. As this application was previously used in the evaluation of the LEON4FT GR740 space-grade microprocessor and XtratuM hypervisor in the context of the EMC² ECSEL project, this will provide an interesting comparison point.

Porting this application to the De-RISC platform will allow us to validate the time and space isolation support that are key requirements for safety and security, and to assess the mechanisms we propose with regards to these properties.

7 Conclusions

Critical space applications have increasing performance, safety, reliability, validation and commercial requirements. While some of them can be attained by adopting available multicore processors, so far no solution reconciles all those requirements in a single product.

De-RISC is a new space platform combining fentISS XtratuM hypervisor and Cobham Gaisler NOEL-V MPSoC meeting all those requirements collaboratively. De-RISC builds upon the non-proprietary RISC-V ISA, and incorporates time predictability and validation technology from BSC. Moreover, De-RISC platform is closely supervised by Thales, a key space and avionics end user, which assesses the platform with a space use case. The De-RISC platform, currently under development, is planned to reach commercialization in early 2022.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement EIC-FTI 869945.

References

[1] D. Andreetti, F. Federici, V. Muttillo, D. Pascucci and L. Pomante, "Analysis and design of a Command & Data Handling platform based on the LEON4 multicore processor and PikeOS hypervisor", 2017. http://hdl.handle.net/11697/137061

[2] J. Bredereke, "A Survey of Time and Space Partitioning for Space Avionics", Technical Report, City University of Applied Sciences Bremen, Bremen, 2016.

[3] J. Cardona, C. Hernandez, J. Abella and F. J. Cazorla, "Maximum-Contention Control Unit (MCCU): Resource Access Count and Contention Time Enforcement," *2019 Design, Automation & Test in Europe Conference & Exhibition* (DATE), Florence, Italy, 2019, pp. 710-715.

[4] Cobham Gaisler, LEON3FT Fault-tolerant processor. https://www.gaisler.com/index.php/products/processors/leon3ft (accessed May-2020)

[5] Cobham Gaisler. LEON4 Processor. https://www.gaisler.com/index.php/products/processors/leon4ft (accessed May-2020).

[6] Cobham Gaisler, "RTEMS SMP Executive Summary, Development Environment for Future Leon Multi-core", *RTEMSSMP-ES-001 issue 2 revision 2*, March 2015 Online: http://microelectronics.esa.int/gr740/RTEMS-SMP-ExecSummary-CGAislerASD-OAR.pdf

[7] The Consultative Committee for Space Data Systems (CCSDS), Lossless multispectral and hyperspectral image compression recommended standard. CCSDS 123.0-B-1. 2012.

[8] FentISS. XtratuM Hypervisor. https://fentiss.com/products/hypervisor/ (accessed May-2020)

[9] B. Gomes, D. Silveira, L. Gouveia, L. Mendes, "Air hypervisor using RTEMS SMP", *European Workshop on On-Board Data Processing* (OBDP2019), ESTEC (ESA), Netherlands, 2019.

[10] J. Jalle et al., "Contention-aware performance monitoring counter support for real-time MPSoCs," *11th IEEE Symposium on Industrial Embedded Systems* (SIES), Krakow, 2016, pp. 1-10.

[11] NXP. e6500 Core Reference Manual. E6500RM Rev 0, 06/2014

[12] RISC-V International. https://riscv.org/risc-v-cores/ (accessed May-2020)

[13] SiFive Inc.. SiFive E76-MC Manual v19.08p0, 2019.

[14] P. K. Valsan, H. Yun and F. Farshchi, "Taming Non-Blocking Caches to Improve Isolation in Multicore Real-Time Systems," *IEEE Real-Time and Embedded Technology and Applications Symposium* (RTAS), Vienna, 2016, pp. 1-12.

[15] Wikipedia. Small satellite. https://en.wikipedia.org/wiki/Small_satellite (accessed May-2020)

Time4PS: Fully Integrated Development Toolset for Partitioned Systems

Yolanda Valiente*, Patricia Balbastre*, Francisco Gómez **, Laurent Rioux, Rafik Henia******

** Universitat Politècnica de València, Spain, **fentISS, Valencia, Spain, ***Thales R&T, Palaiseau, France*

Abstract

In this paper, we propose an architecture for the design and deployment of real-time partitioned systems called Time4PS. Time4PS bridge the gap between modelling tools and deployment tools so the design, modelling, analysis and integration can be done in a unique framework.

Keywords: partitioned systems, modelling.

1 Introduction

Partitioned systems are the most promising technology to build dependable systems where some applications are executed in the same hardware platform. In a partitioned system, many different applications are integrated onto the same hardware each executing in their own partition. They are kept separated by the partitioning kernel, such as the XtratuM[1] hypervisor (Figure 1).

In this kind of systems, the allocation of resources is made statically through a configuration file. These resources are the memory allocation, the scheduling plan, communication between partitions, IO ports, etc. Editing the XtratuM configuration file as plain text is a complex task that is subject to errors and inconsistencies due to interdependence of information and the editing process itself. This is why it is convenient to use a tool that facilitates the process of generating the configuration file. The Xamber [2] graphical configuration tool was developed to ease this tedious and laborious task. Furthermore, a modelling design tool (different depending on the customer) is used to model the system and generate automatic code of the applications. The modelling tool is often used by our customers to design the entire system, while the configuration tool is used to assign the available resources. There is a gap between these two tools. We need an intermediate tool to ease this process: Time4Sys[3], which will play the role of connecting and bridge the identified gap by providing pivot models which are independent of any design tool. The integration of XtratuM tools in a full software development lifecycle through Time4Sys will enable customers to use more sophisticated tools for modelling, analysis and integration of their system and it will give us the ability to develop further support of their products in Time4Sys and to offer their expertise in this tool to interested customers.

Industrial Presentation submitted for the Ada-Europe International Conference (AEiC 2020).

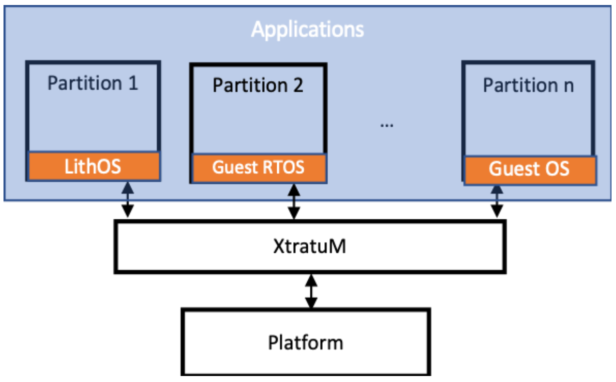


Figure 1. Partitioned architecture with XtratuM

2 Solution

Time4PS aims at providing a complete set of tools that covers all the phases of the life cycle of the product: from the system design to the final configuration file of the partitioned system. Time4PS toolset consist of 3 connected tools:

- EEI/XPM: a modelling and deployment tool for XtratuM. Along with the temporal model, XPM enables the user to define a resource model which groups all other system elements which are required by the XtratuM architecture. Mainly, this resource system model is related with the strong spatial isolation, the communication and the Input and Output. Moreover, XPM automatically generates the build and deployment scripts for a target partitioned system based on the XtratuM hypervisor.
- Xamber: a configuration tool that assigns temporal and spatial resources and verifies the correctness of a partitioned system. Specifically, this application is responsible for:
    - simplifying resource allocation: allocation of the system resources (memory, peripherals, interrupts, etc.) and inter-partition communication (connection of ports) can be provided in a graphical way.
    - provide temporal allocation, that is: schedule plan generation and the schedulability analysis based on the temporal model defined in Time4Sys.

  - verification: detection of inconsistencies in the resource allocation and temporal allocation.
- Time4Sys: a timing performance framework. It can be connected to various environments and languages such as UML, SysML, AADL, or any other proprietary environment (e.g. Capella).

The solution proposed can be seen Figure 2, where Time4PS architecture is depicted. The goal is to close the loop from the definition of a partitioned meta-model to the generation of the final static configuration file with which the system can be executed.

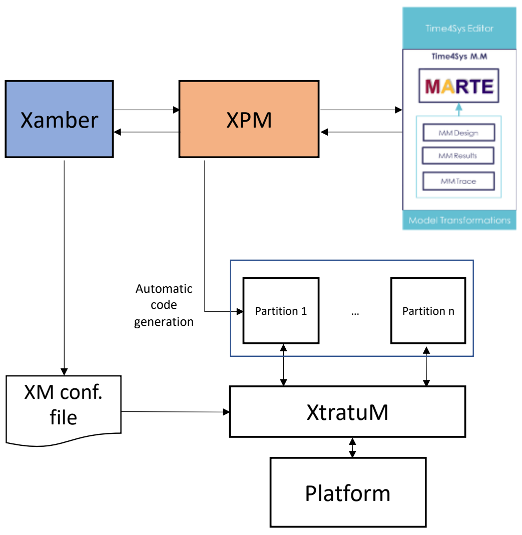


Figure 2. Time4PS architecture

2.1 Workflow

We can see that EEI/XPM is the central tool that manages all the information. This is due to the fact that it is responsible of the generation of the final configuration file needed for the deployment of the system. In order to generate this file, all three tools must work together.

The workflow of the toolset is as follows:

1. Define the temporal model in Time4Sys and the non-temporal model in EEI/XPM.
2. Import the Time4Sys model to EEI/XPM.
3. Export the temporal model (the functional model is not needed for the configuration generation) from the EEI/XPM to Xamber. Allocate resources, generate the scheduling plan and verify the correctness. If correct, go to next step, else go back to the step 1.
4. Export the Xamber data to EEI/XPM.
5. Export the complete temporal model from EEI/XPM to Time4Sys. Verify in time4sys that all applications functions meet their deadline. If correct, go to 6, if not go back to 1.
6. Finally, generate the final configuration file for XtratuM and deploy the final system

The work to be done involves the modification of Time4Sys (extensions to support the ARINC653 standard) and Xamber to be connected to EEI/XPM and the re-design and integration of two fentISS tools: EEI and XPM. EEI was a tool developed under an ESA project called IMA-SDT.

EEI is the tool in charge of the definition of the system model while XPM tool is developed under the MegaM@RT project to deploy a partitioned system. The integration of these two tools reduces the number of tools needed to create this kind of systems.

2.2 EEI/XPM design

EEI/XPM is the central tool that concentrates all the connections between others. It is developed as an Eclipse plugin. Most of the effort has been done in converting Time4Sys and Xamber to EEI/XPM model.

EEI/XPM is developed as an Eclipse plugin. The components of the EEI/XPM and the connection between the tools involved in the Time4PS architecture are depicted in Figure 3.

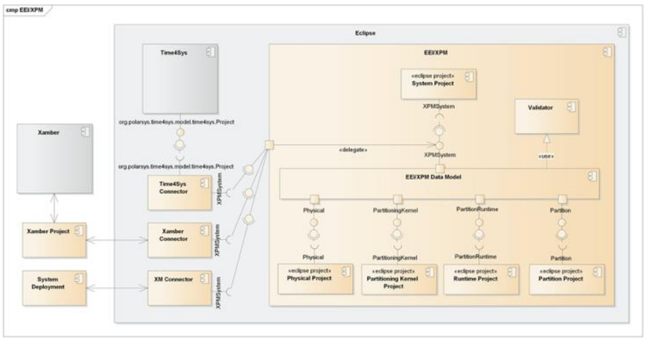


Figure 3. EEI/XPM components

The main features of EEI/XPM are:

- Edits and validates the EEI/XPM Data Model, which defines the data model of a partitioned system based on the XtratuM hypervisor. The Validator component performs the validation of the data model.
- Provides modularity. The data model is divided in several parts and new eclipse projects are introduced in the Eclipse IDE to manage them. Each new eclipse project can be treated as a common Eclipse project, that is, it can be imported in Eclipse for reuse through Eclipse main menu Import... > General > Existing Project into Workspace. It provides the modularity which helps to use already existent model parts.
- The Time4Sys Connector allows the connection between the Time4Sys framework and the tool.
- The Xamber Connector allows the connection between the EEI/XPM and the Xamber tool though import/export of a Xamber Project, which can be loaded/saved by Xamber.
- Generates the deployment of the system (System Deployment), where all partitions are integrated to build the final system on a specific target. The

deployment generation is performed by the XM Connector for a specific version of the XtratuM hypervisor. There are as many XM Conectors as versions of XtratuM supported.

- Imports an existing XtratuM Configuration File (XCF/XMCF). This is also performed by the XM Connector.

2.3 EEI/XPM data model

The data model proposed of a partitioned system is depicted in Figure 4.

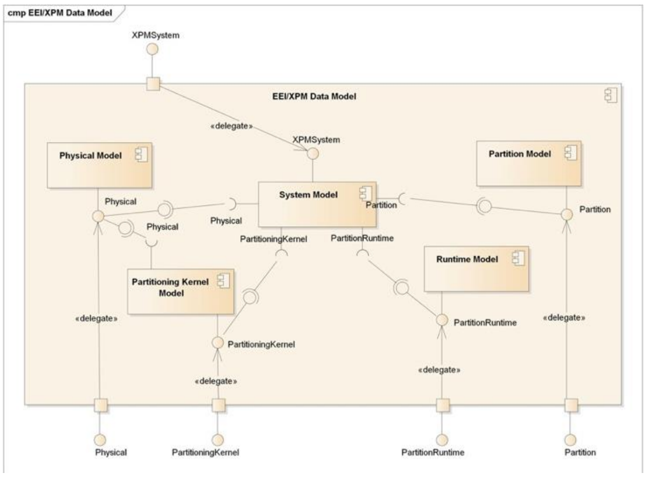


Figure 4. EEI/XPM data model

The EEI/XPM data model defines a model for each elemental part of a system based on the XtratuM hypervisor:

- Physical: The target's board Board-Support Package (BSP) where the deployment is performed. It describes the Hardware Resources (processor, memory, hardware devices...) that are available for the partitions.
- Partitioning Kernel: The XtratuM hypervisor description including version, the physical configuration for the hypervisor and technical characteristics. Moreover, it includes the location to the XtratuM hypervisor distribution package, which provides to the plug-in relevant information for building the system.
- Runtime: The execution environment that can be used by partitions: a Run-Time Environment (table) or the hypervisor. When appropriate, the location to the runtime is included.
- Partition: The program unit definition and the partition resource requirements.
- System: The configuration of the system. It provides the physical configuration, the partitions of the system and their runtimes, and the hypervisor to perform the system deployment. These elements allow to perform the resources allocation in Xamber in order to fulfill the requirements defined.

2.4 Time4Sys connector

The Time4Sys connector is able to bridge the EEI/XPM tool to the Time4sys framework, as depicted in Figure 5. This connector will allow exporting a Time4Sys model into an EEI/XPM model and vice versa. Specifically:

- Imports a Time4Sys model. This connector converts data from a Time4sys model to an existent EEI/XPM data model, specifically to the temporal model using mappings.
- Exports to a Time4Sys model from a full or partial EEI/XPM data model. This connector converts data from the temporal model of an EEI/XPM data model to an existent Time4Sys model, using mappings. Moreover, the connector will allow importing the complete temporal model back into Time4Sys.

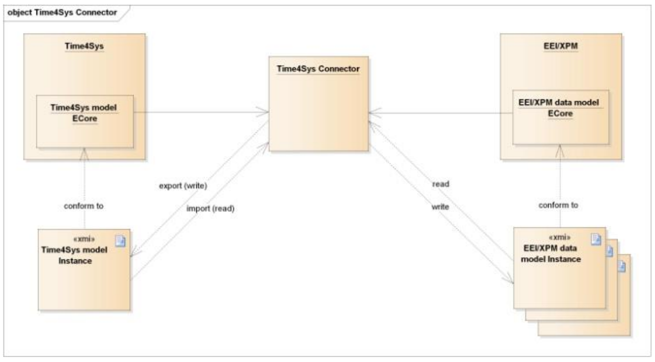


Figure 5. Time4Sys connector

Both the Time4Sys framework and the EEI/XPM tool store their model content via the EMF persistence framework in XMI, which is the serialization format used for representing their metamodels (Ecore). The Time4Sys model Ecore and the EEI/XPM data model Ecore show their root object, org.polarsys.time4sys.model.time4sys.Project and XPMSystem respectively, representing the whole model, which are required to the Time4Sys connector, for performing the models conversion.

2.5 Xamber connector

The Xamber connector is able to bridge the EEI/XPM tool to the Xamber tool, as depicted in Figure 6. This connector will allow exporting an EEI/XPM data model model into a Xamber Project and vice versa.

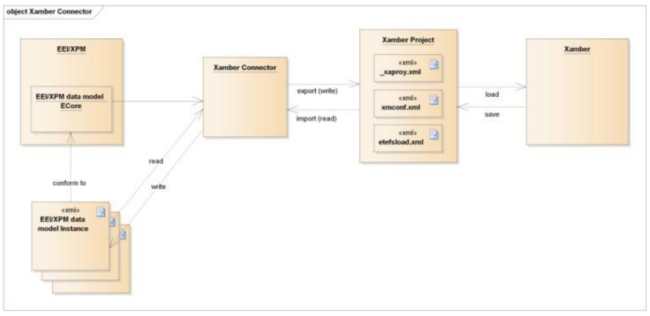


Figure 6. Xamber connector

The EEI/XPM tool stores its model content via the EMF persistence framework in XMI, which is the serialization format used for representing the metamodel, EEI/XPM data model Ecore. The Ecore shows the root object XPMSystem representing the whole model, which is required to the Time4Sys connector, for performing the conversion from an EEI/XPM data model to a Xamber Project.

Xamber generates Java classes that mirror the Xamber model from the XML Schema (XSD). This is done using Java Architecture for XML Binding (JAXB) binding compiler xjc. The JAXB binding compiler takes XML schema as input, and then generates a package of Java classes and interfaces that reflect the rules defined in the source schema. The Java classes generated with the JAXB binding compiler xjc utility represent the different elements and complexType(s) in an XML Schema. An XML document that conforms to the XML Schema may be constructed from the Java classes. Xamber saves the Xamber model content in a project made up of several files in XML format.

3 Conclusions

In the development of partitioned systems two tools are used to define the complete system. A modelling design tool that is used to design the system and to generate automatic code of the applications and a configuration tool to obtain the static configuration file. There is a gap between the two tools. The modelling tool is often used by the customers to define the entire system, while the configuration tool (Xamber) is used by the system integrator to assign the available resources. Time4Sys plays the role of connecting and bridge the identified gap. The proposed toolset is called Time4PS, partially developed with FED4SAE funding. The toolset will be validated with a use-case provided by Thales: The Flight Management System (FMS) [4]. Time4PS will empower owners to gain global competitiveness in three ways. Firstly, it will allow to enter in customers who are already using Time4Sys or UML/MARTE models. Secondly, it will improve the company products position in large and complex projects who use a more sophisticated software development lifecycle in which Time4Sys is integrated thus allowing schedulability analysis and timing simulation in early stages of software development. Finally, the developed toolset will offer new opportunities to be used in adjacent markets such as avionics, railways or automotive.

Acknowledgements

This work is done partially under the support of FED4SAE projects (grant agreement No 761708), MegaM@RT ECSEL JU project (grant agreement No 737494) and Agencia Estatal de Investigación (TIN2017-86520-C3-1-R-AR).

References

[1] E. Carrascosa, J. Coronel, Javier; M. Masmano, P. Balbastre Betoret, A. Crespo, "XtratuM hypervisor redesign for LEON4 multicore processor", *ACM SIGBED Review, 2* (11), 27 – 31, 2014.

[2] J.Simo, Y. Valiente, P. Balbastre, "Xamber User Manual v2.2", September 2019.

[3] Time4Sys. https://www.eclipse.org/time4sys/

[4] THALES, "FMS220 Software Requirement Specification (SRS)", Technical report, THALES, Nov 2010.

The problem of the Greek Cross

John Barnes

11 Albert Road, Caversham, Reading, RG4 7AN, UK; Tel: +44 118 9474125; email: john@jbinformatics.co.uk

Hello readers

I thought I should mention some other puzzles from old conferences just for the record. But I will start by giving the answers to the puzzles mentioned last time.

The nasty multiplication is

```
          1  7  9
          2  2  4
-----------------
3  5  8
   3  5  8
      7  1  6
-----------------
4  0  0  9  6
```

Remember that all we were told is that each of the ten digits from 0 to 9 occurs exactly twice.

The other puzzles were of the type where digits are replaced by letters. Leading zeroes are of course forbidden. The puzzles with answers beneath are

```
HOCUS + POCUS = PRESTO
92836    + 12836   = 105672

COUPLE + COUPLE = QUARTET
653924    + 653924     = 1307848

ZEROES + ONES = BINARY
698392    + 3192   = 701584

FISH + N + CHIPS = SUPPER
5718  + 3  + 98741 = 104462
```

and finally with multiplication rather than addition

```
PI * R * R = AREA
96 * 7 * 7 = 4704
```

Such puzzles can be solved by trial and error but since we are Ada folk perhaps the use of an Ada program is permitted.

The key trick is to generate all permutations of the relevant letters. This can be done by the following procedure

```
type Atype is array(Integer range <>) of Integer;
procedure Generate(K: Integer; A: in out Atype) is
  Temp: Integer;
begin
  if K = 1 then
    Check;
  else
    for I in 0 .. K-1 loop
      Generate(K-1, A);
      if K mod 2 = 0 then              -- K even
        Temp := A(I);  A(I) := A(K-1);  A(K-1) := Temp;
      else                   -- K odd
        Temp := A(0);  A(0) := A(K-1);  A(K-1) := Temp;
      end if;
    end loop;
  end if;
end Generate;
```

Now we declare an array X thus

```
X: Atype(0 .. 9) := (0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
```

and call Generate with

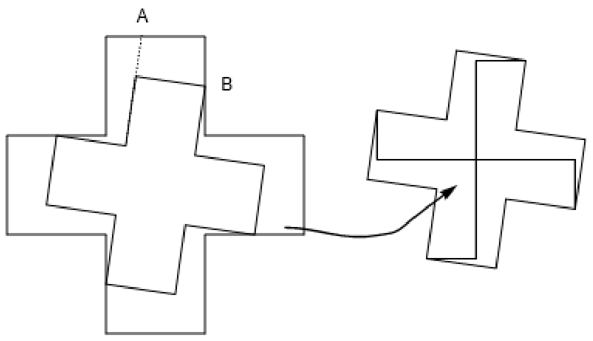
```
Generate(10, X);
```

This causes Check to be called with all the possible permutations in X. So, we need to write the procedure Check to see whether the current permutation in X solves the problem.

The permutation algorithm was devised by B R Heap in 1963. There is a good article in Wikipedia.

And now here is a rather different puzzle. Can you divide a Greek cross into five pieces that fit together to form two equal smaller Greek crosses. The answer is Yes but how?

The diagram below on the left shows the original cross with a small cross at an angle inside. The four pieces around the



small cross are removed and form the cross on the right.

The question is where on the short sides of the original cross are the points A and B which determine where the cuts are made?

This puzzle appears in *Amusements in Mathematics* by H E Dudeney published in 1917. He does not give the answer. The puzzle also appears in the *Boy's Own Paper* for December 1917.

The answer given in the next issue of the *BOP* is that B is midway along the short side (that is correct). It also says that A is one-third of the way along the end. That is wrong. What is the correct answer? Where is A?

There are tedious solutions but also a cunning easy one.

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

Ada Edge

27 Rue Rasson
B-1030 Brussels, Belgium
Contact:Ludovic Brenta
ludovic@ludovic-brenta.org

AdaCore
The GNAT Pro Company

46 Rue d'Amsterdam
F-75009 Paris, France
Contact: Jamie Ayre
sales@adacore.com
www.adacore.com

ADA LOG

2 Rue Docteur Lombard
92441 Issy-les-Moulineaux Cedex
France
Contact: Jean-Pierre Rosen
rosen@adalog.fr
www.adalog.fr/en/

altran

22 St. Lawrence Street
Southgate
Bath BA1 1AN, United Kingdom
Contact: Stuart Matthews
sparkinfo@altran.com
www.altran.co.uk

Deep Blue Capital

Jacob Bontiusplaats 9
1018 LL Amsterdam
The Netherlands
Contact: Wido te Brake
wido.tebrake@deepbluecap.com
www.deepbluecap.com

Ellidiss Technologies

24 Quai de la Douane
29200 Brest, Brittany
France
Contact: Pierre Dissaux
pierre.dissaux@ellidiss.com
www.ellidiss.com

KONAD
Software for Control and Administration

In der Reiss 5
D-79232 March-Buchheim
Germany
Contact: Frank Piron
info@konad.de
www.konad.de

PTC® Developer Tools

3271 Valley Centre Drive,
Suite 300
San Diego, CA 92069, USA
Contact: Shawn Fanning
sfanning@ptc.com
www.ptc.com/developer-tools

SYSADA

Signal Business Centre
2 Innotec Drive, Bangor
North Down BT19 7PD
Northern Ireland, UK
enquiries@sysada.co.uk
www.sysada.co.uk

systerel
Safe real-time solutions

1090 Rue René Descartes
13100 Aix en Provence, France
Contact: Patricia Langle
patricia.langle@systerel.fr
www.systerel.fr/en/

Tid rum

Tiirasaarentie 32
FI 00200 Helsinki, Finland
Contact: Niklas Holsti
niklas.holsti@tidorum.fi
www.tidorum.fi

VECTOR

Millennium Tower, floor 41
Handelskai 94-96
A-1200 Austria
Contact: Massimo Bombino
sales@at.vector.com
www.vector.com

WhiteElephant

Beckengässchen 1
8200 Schaffhausen
Switzerland
Contact: Ahlan Marriott
admin@white-elephant.ch

XGC Technology

United Kingdom
Contact: Chris Nettleton
nettelton@xgc.com
www.xgc.com

Ada
europe