ADA USER JOURNAL

Volume 41
Number 1
March 2020

Contents

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.
—arm]

Ada-related Events

25th Ada-Europe Int'l Conf. on Reliable Software Technologies

[This year's edition has been cancelled (see cancellation post below). This extended call is reproduced here for reference. —arm]

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Subject: Ada-Europe 2020 Conference - EXTENDED 14 January deadline
Date: Thu, 19 Dec 2019 18:29:46 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

The Ada-Europe 2020 Conference organizers decided to provide more time for authors to prepare their contributions. The deadline for most submissions is extended to Tuesday 14 January 2020. 3+ weeks remain!

Call for Contributions

UPDATED Call for Papers - EXTENDED DEADLINE

25th Ada-Europe International Conference on
Reliable Software Technologies
(AEiC 2020)

8-12 June 2020, Santander, Spain

www.ada-europe.org/conference2020

Organized by University of Cantabria and Ada-Europe
in cooperation with ACM SIGAda (pending)

and the Ada Resource Association (ARA)

*** Extended DEADLINE
14 JANUARY 2020 AoE ***

#AdaEurope #AEiC2020
#AdaProgramming

General Information

The 25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2020 aka Ada-Europe 2020) will take place in Santander, Spain, in the week of 8-12 June. The conference schedule includes a technical program and vendor exhibition, and parallel tutorials and workshops.

The 2020 edition of the conference continues the major revamp in the registration fees introduced in 2019, redesigned to extend participation from industry and academia, and to reward contributors, especially but not solely, students and post-doc researchers.

Schedule

14 January 2020: Submission of journal-track papers, industrial presentation outlines, and tutorial and workshop proposals (EXTENDED)

20 March 2020: Notification of acceptance for journal-track papers, industrial presentations, tutorials and workshops

31 March 2020: Submission of Work-in-Progress (WiP) papers

30 April 2020: Notification of acceptance for WiP papers

Topics

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

The topics of interest for the conference include but are not limited to:

- Design and Implementation of Real-Time and Embedded Systems: Real-Time Scheduling, Design Methods and Techniques, Architecture Modelling, HW/SW Co-Design, Reliability and Performance;
- Design and Implementation of Mixed-Criticality Systems: Scheduling Methods, Mixed-Criticality Architectures, Design Methods, Analysis Methods;
- Theory and Practice of High-Integrity Systems: Medium to Large-Scale Distribution, Fault Tolerance, Security, Reliability, Trust and Safety, Languages Vulnerabilities;
- Software Architectures for Reliable Systems: Design Patterns, Frameworks, Architecture-Centered Development, Component-based Design and Development;
- Methods and Techniques for Quality Software Development and Maintenance: Requirements Engineering, Model-driven Architecture and Engineering, Formal Methods, Re-engineering and Reverse Engineering, Reuse, Software Management Issues, Compilers, Libraries, Support Tools;
- Ada Language and Technologies: Compilation Issues, Runtimes, Ravenscar, Profiles, Distributed Systems, SPARK;
- Mainstream and Emerging Applications with Reliability Requirements: Manufacturing, Robotics, Avionics, Space, Health Care, Transportation, Cloud Environments, Smart Energy Systems, Serious Games, etc;
- Achieving and Assuring Safety in Machine Learning Systems;
- Experience Reports in Reliable System Development: Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics;
- Experiences with Ada: Reviews of the Ada 2012 language features, implementation and use issues, positioning in the market and in the software engineering curriculum, lessons learned on Ada Education and Training Activities with bearing on any of the conference topics.

Call for Journal-Track Papers

The journal-track papers submitted to the conference are full-length papers that must describe mature research work on the conference topics. They must be original and shall undergo anonymous peer review.

Accepted journal-track papers will get a presentation slot within a technical session of the conference and they will be published in an open-access special issue of the Journal of Systems Architecture with no additional costs to authors.

The corresponding authors shall submit their work by 14 January 2020 via the Special Issue web page: https://www.journals.elsevier.com/journal-of-systems-architecture/call-for-papers/advances-in-reliable-software-technologies

Submitted papers must follow the guidelines provided in the "Guide-for-Authors" of the JSA (https://www.elsevier.com/journals/journal-of-systems-architecture/1383-7621/guide-for-authors). In particular, JSA does not impose any restriction on the format or extension of the submissions.

Call for WiP-Track Papers

The Work-in-Progress papers (WiP-track) are short (4-page) papers describing evolving and early-stage ideas or new research directions. They must be original and shall undergo anonymous peer review. The corresponding authors shall submit their work by 31 March 2020, via https://easychair.org/conferences/?conf=aeic2020, strictly in PDF and following the Ada User Journal style (http://www.ada-europe.org/auj/).

Authors of accepted WiP-track papers will get a presentation slot within a regular technical session of the conference and will also be requested to present a poster. The papers will be published in the Ada User Journal as part of the proceedings of the Conference.

The conference is listed in the principal citation databases, including DBLP, Scopus, Web of Science, and Google Scholar. The Ada User Journal is indexed by Scopus and by EBSCOhost in the Academic Search Ultimate database.

Call for Industrial Presentations

The conference seeks industrial presentations that deliver insightful information value but may not sustain the strictness of the review process required for regular papers. The authors of industrial presentations shall submit their proposals, in the form of a short (one or two pages) abstract, by 14 January 2020, via https://easychair.org/conferences/?conf=aeic2020, strictly in PDF and following the Ada User Journal style (http://www.ada-europe.org/auj/).

The Industrial Committee will review the submissions anonymously and make recommendations for acceptance. The abstract of the accepted contributions will be included in the conference booklet, and authors will get a presentation slot within a regular technical session of the conference.

These authors will also be invited to expand their contributions into articles for publication in the Ada User Journal, as part of the proceedings of the Industrial Program of the Conference.

Awards

Ada-Europe will offer an honorary award for the best presentation.

Call for Educational Tutorials

The conference is seeking tutorials in the form of educational seminars including hands-on or practical demonstrations. Proposed tutorials can be from any part of the reliable software domain, they may be purely academic or from an industrial base making use of tools used in current software development environments. We are also interested in contemporary software topics, such as IoT and artificial intelligence and their application to reliability and safety.

Tutorial proposals shall include a title, an abstract, a description of the topic, an outline of the presentation, the proposed duration (half day or full day), and the intended level of the tutorial (introductory, intermediate, or advanced). All proposals should be submitted by e-mail to the Educational Tutorial Chair.

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

Call for Workshops

Workshops on themes that fall within the conference scope may be proposed. Proposals may be submitted for half- or full-day events, to be scheduled at either end of the conference days. Workshop proposals should be submitted by e-mail to the Workshop Chair. The workshop organizer shall also commit to producing the proceedings of the event, for publication in the Ada User Journal.

Call for Exhibitors

The commercial exhibition will span the core days of the main conference. Vendors and providers of software products and services should contact the Exhibition Chair for information and for allowing suitable planning of the exhibition space and time.

Special Registration Fees

Authors of accepted contributions and all students will enjoy reduced registration fees.

Venue

Santander is a nice tourist city in the north of Spain, with a well-connected airport and at a 100 km drive from Bilbao airport.

The conference venue and hotel is the Bahia Hotel in the city center and beside Santander bay.

Organizing Committee

* Conference Chair

Michael González Harbour, Universidad de Cantabria, Spain

mgh at unican.es

* Program Chair

Mario Aldea Rivas, Universidad de Cantabria, Spain

aldeam at unican.es

* Work-in-Progress Chair

Kristoffer Nyborg Gregertsen, SINTEF Digital, Norway

kristoffer.gregertsen at sintef.no

* Tutorial & Workshop Chair

Jorge Garrido Balaguer, Universidad Politécnica de Madrid, Spain

jorge.garrido at upm.es

* Industrial Chair

Patricia Balbastre Betoret, Universitat Politècnica de València, Spain

patricia at ai2.upv.es

* Exhibition & Sponsorship Chair

Ahlan Marriott, White Elephant GmbH, Switzerland

software at white-elephant.ch

* Publicity Chair

Dirk Craeynest, Ada-Belgium & KU Leuven, Belgium

dirk.craeynest at cs.kuleuven.be

*** Program Committee

Mario Aldea Rivas, Univ. de Cantabria, ES

Iain Bate, University of York, UK

Johann Blieberger, Vienna Univ. of Technology, AT

Bernd Burgstaller, Yonsei Univ., KR

Daniela Cancila, CEA LIST, FR

António Casimiro, Univ. Lisboa, PT

Juan A. de la Puente, Univ. Pol. de Madrid, ES

Barbara Gallina, Mälardalen Univ., SE

Marisol García Valls, Univ. Pol. de València, ES

J. Javier Gutiérrez, Univ. de Cantabria, ES

Jérôme Hugues, CMU/SEI (USA)

Hubert Keller, Karlsruhe Institute of Technology, DE

Patricia López Martínez, Univ. de Cantabria, ES

Kristoffer Nyborg Gregertsen, SINTEF Digital, NO

Laurent Pautet, Telecom ParisTech, FR

Luís Miguel Pinho, CISTER/ISEP, PT

Erhard Plödereder, Univ. Stuttgart, DE

Jorge Real, Univ. Pol. de València, ES

José Ruiz, AdaCore, FR

Sergio Sáez, Univ. Pol. de València, ES

Frank Singhoff, Univ. de Bretagne Occidentale, FR

Tucker Taft, AdaCore, USA

Elena Troubitsyna, Åbo Akademi Uni., FI

Santiago Urueña, GMV, ES

Tullio Vardanega, Univ. of Padua, IT

Eugenio Villar Bonet, Univ. de Cantabria, ES

Industrial Committee

Ian Broster, Rapita Systems, UK

Javier Coronel, FentISS, ES

Dirk Craeynest, Ada-Belgium & KU Leuven, BE

Thomas Gruber, Austrian Institute of Technology (AIT), AT

Ismael Lafoz, Airbus Defence and Space, ES

Ahlan Marriott, White Elephant, CH

Maurizio Martignano, Spazio IT, IT

Laurent Rioux, Thales R&T, FR

Marian Roselló, Terma, NL

Jean-Pierre Rosen, Adalog, FR

Emilio Salazar, GMV, ES

Previous Editions

Ada-Europe organizes annual international conferences since the early 80's. This is the 25th event in the Reliable Software Technologies series, previous ones being held at Montreux, Switzerland ('96), London, UK ('97), Uppsala, Sweden ('98), Santander, Spain ('99), Potsdam, Germany ('00), Leuven, Belgium ('01), Vienna, Austria ('02), Toulouse, France ('03), Palma de Mallorca, Spain ('04), York, UK ('05), Porto, Portugal ('06), Geneva, Switzerland ('07), Venice, Italy ('08), Brest, France ('09), Valencia, Spain ('10), Edinburgh, UK ('11), Stockholm, Sweden ('12), Berlin, Germany ('13), Paris, France ('14), Madrid, Spain ('15), Pisa, Italy ('16), Vienna, Austria ('17), Lisbon, Portugal ('18), and Warsaw, Poland ('19).

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

Our apologies if you receive multiple copies of this announcement. Please circulate widely.

Dirk.Craeynest@cs.kuleuven.be, Ada-Europe 2020 Publicity Chair

*** 25th Ada-Europe Int'l. Conf. on Reliable Software Technologies ***

June 8-12, 2020 * Santander, Spain * www.ada-europe.org/conference2020

Ada-Europe Int'l Conference 2020 (AEiC 2020) Cancelled!

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Subject: Ada-Europe Int'l Conference 2020 (AEiC 2020) cancelled!
Date: Sat, 21 Mar 2020 20:15:38 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

[Notice of Cancellation is included in the Forthcoming Events Section —arm]

10th Ada Developer Room at FOSDEM 2020 - Summary of Talks

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Subject: FOSDEM 2020 - Ada Developer Room - Sat 1 Feb 2020 - Brussels
Date: Sun, 22 Dec 2019 21:53:32 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

Ada-Belgium is pleased to announce its

10th Ada Developer Room at FOSDEM 2020

Ada at the Free and Open source Software Developers' European Meeting on Saturday 1 February 2020

Université Libre de Bruxelles (ULB), Solbosch Campus, Room AW1.125

Avenue Franklin D. Roosevelt Laan 50, B-1050 Brussels, Belgium

Organized in cooperation with Ada-Europe

www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem.html

fosdem.org/2020/schedule/track/ada

General Information

FOSDEM, the Free and Open source Software Developers' European Meeting, is a free and non-commercial two-day weekend event organized early each year in Brussels, Belgium. It is highly developer-oriented and brings together 8000+ participants from all over the world.

The goal is to provide open source developers and communities a place to meet with other developers and projects, to be informed about the latest developments in the open source world, to attend interesting talks and presentations on various topics by open source project leaders and committers, and to promote the development and the benefits of open source solutions.

The 2020 edition takes place on Saturday 1 and Sunday 2 February. It is free to attend and no registration is necessary.

In this edition, Ada-Belgium organizes once more a series of presentations related to the Ada Programming Language and Free or Open Software in a s.c. Developer Room. The "Ada DevRoom" at FOSDEM 2020 is held on the first day of the event, Saturday 1 February 2020.

This year FOSDEM has a total of 13 Ada-related presentations by 12 authors from 8 countries! A mini-poster about the Ada DevRoom [1], as well as a one-page Call for Participation for the Ada DevRoom [2] is available; they can be used to help announce the event, and to give an idea about its scope.

[1] www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem-cfpart-poster.jpg

[2] www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem-cfpart-a4.pdf

Ada Programming Language and Technology

Ada is a general-purpose programming language originally designed for safety- and mission-critical software engineering. It is used extensively in air traffic control, rail transportation, aerospace, nuclear, financial services, medical devices, etc. It is also perfectly suited for open source development.

Awareness of safety and security issues in software systems is ever increasing. Multi-core platforms are now abundant. These are some of the reasons that the Ada programming language and technology attracts more and more attention, among others due to Ada's support for programming by contract and for multi-core targets. The latest Ada language definition was updated early 2016. Work on new features is ongoing, such as improved support for fine-grained parallelism, and will result in a new Ada standard scheduled for 2021. Ada-related technology such as SPARK provides a solution for the safety and security aspects stated above. More and more tools are available, many are open source, including for small and recent platforms. Interest in Ada keeps further increasing, also in the open source community, and many exciting projects have been started.

The Ada DevRoom aims to present the facilities offered by the Ada language (such as for object-oriented, multicore, or embedded programming) as well as some of the many exciting tools and projects using Ada. FOSDEM is an ideal fit for an Ada Developer Room. On the one hand, it gives the general open source community an opportunity to see what is happening in the Ada community and how Ada

technology can help to produce reliable and efficient open source software. On the other hand, it gives open source Ada projects an opportunity to present themselves, get feedback and ideas, and attract participants to their project and collaboration between projects.

Video/Volunteers

This year as well, audio/video equipment and network facilities are provided by the FOSDEM organizers, to enable recording and live streaming all DevRoom presentations. Volunteers "man" that equipment during the day. After postprocessing the recordings, links to them are made available via the "More information" entry for each presentation.

Additional volunteers to help with various logistic issues during the day are needed as well, such as monitoring room overflow and refusing entry when the room is too full, defragmenting the room in between presentations, helping speakers with microphone adjustments, monitoring the timeslots and warning speakers when they have to start or when they risk running out of time, as well as various practical issues that need to be handled ASAP when they occur.

If you'd like to help, please get in touch (see below).

Ada Developer Room Presentations (room: AW1.125, 76 seats)

The presentations in the Ada DevRoom start after the opening FOSDEM keynotes. The program runs from 10:30 to 19:00.

10:00-10:30 - Arrival & Informal Discussions

Feel free to arrive early, to start the day with some informal discussions while the set-up of the DevRoom is finished.

10:30-10:35 - Welcome to the Ada DevRoom by Dirk Craeynest - Ada-Belgium

Welcome to the Ada Developer Room at FOSDEM 2020, which is organized by Ada-Belgium in cooperation with Ada-Europe. Ada-Belgium and Ada-Europe are non-profit organizations set up to promote the use of the Ada programming language and related technology, and to disseminate knowledge and experience into academia, research and industry in Belgium and Europe, resp. Ada-Europe has member-organizations, such as Ada-Belgium, in various countries, and direct members in many other countries.

10:35-11:20 - An Introduction to Ada for Beginning and Experienced Programmers by Jean-Pierre Rosen - Adalog, France

An overview of the main features of the Ada language, with special emphasis on those features that make it especially attractive for free software development. Ada is a feature-rich language, but what really makes Ada stand-out is that the features are nicely integrated towards serving the goals of software engineering. If you prefer to spend your time on designing elegant solutions rather than on low-level debugging, if you think that software should not fail, if you like to build programs from readily available components that you can trust, you should really consider Ada!

11:30-11:50 - HAC: the Compiler which will Never Become Big by Gautier de Montmollin - Ada-Switzerland

In the Ada world, we are surrounded by impressive and professional tools that can handle large and complex projects. Did you ever dream of a tiny, incomplete but compatible system to play with? Are you too impatient, for developing small pieces of code, for long compile-bind-link-run cycles? Are you a beginner intimidated by project files and sophisticated tools? Then HAC (the HAC Ada Compiler, or the Hello-world Ada Compiler) is for you. HAC is a revival of the SmallAda project, which supported the "Pascal subset" plus tasking.

12:00-12:50 - Tracking Performance of a Big Application from Dev to Ops by Philippe Waroquiers - Eurocontrol, Belgium

This talk describes how performance aspects of a big Air Traffic Flow Management mission critical application are tracked from development to operations. Tracking performance is needed when new functionality is added, to balance the additional services versus the resource increase needed. Measuring and tracking performance is also critical to ensure a new release can cope with the current or expected load. We will discuss various aspects such as which tools and techniques are used for performance tracking and measurements, what are the traps and pitfalls encountered for these activities. The application in question is using Ada, but most of the items discussed are not particularly Ada related.

13:00-13:20 - Cappulada: What we've Learned by Johannes Kliemann - Componolit, Germany

Last year I presented Cappulada, a C++ binding generator for Ada that intended to overcome the shortcomings of existing solutions and to provide usable bindings even for complex C++ code. This year I want to show our conclusions on why automatic bindings between C++ and Ada are hard (if not impossible) and where existing solutions (including our own) fail.

13:30-13:50 - Programming ROS2 Robots with RCLAda by Alejandro R. Mosteo - Centro Universitario de la Defensa, Spain

The Robot Operating System (ROS) is one of the chief frameworks for service robotics research and development. The next iteration of this framework, ROS2, aims to improve critical shortcomings of its predecessor like deterministic memory allocation and real-time characteristics. RCLAda is a binding to the ROS2 framework that enables the programming of ROS2 nodes in pure Ada with seamless integration into the ROS2 workflow.

14:00-14:50 - Live Demo of Ada's Distribution Features by Jean-Pierre Rosen - Adalog, France

Ada incorporates in its standard a model for distributed execution. It is an abstract model that does not depend on a particular kind of network or any other communication mean, and that preserves full typing control across partitions. This presentation briefly exposes the principles of Ada's distribution model, then shows the possibilities with life demos across different machines and operating systems.

15:00-15:20 - Writing Shared Memory Parallel Programs in Ada by Jan Verschelde - University of Illinois at Chicago, USA

Multitasked Newton's Method for Power Series Tasks in Ada are effective to speed up computations on multicore processors. In writing parallel programs we determine the granularity of the parallelism with respect to the memory management. We have to decide on the size of each job, the mapping of the jobs to the tasks, and on the location of the input and output data for each job. A multitasked Newton's method will show the effectiveness of Ada to speed up the computation of power series. This application belongs to the free and open source package PHCpack, a package to solve polynomial systems by polynomial homotopy continuation.

15:30-15:50 - Spunky: a Genode Kernel in Ada/SPARK by Martin Stein - Genode Labs, Germany

The Genode OS framework is an open-source tool kit for building highly secure component-based operating systems scaling from embedded devices to dynamic desktop systems. It runs on a variety of microkernels like SeL4, NOVA, and Fiasco OC as well as on Linux and the Muen SK. But the project also features its own microkernel named "base-hw" written in C++ like most of the Genode framework. Spunky is a pet project of mine. Simply put it's an approach to re-implement the design of the "base-hw" kernel first in Ada and later in SPARK with the ultimate goal to prove its correctness. It is also an opportunity to learn how Genode can benefit from Ada and SPARK in general and promote the use of safety-oriented languages in the project.

16:00-16:50 - Alire: Ada Has a Package Manager by Alejandro R. Mosteo - Centro Universitario de la Defensa, Spain, Pierre-Marie de Rodat and Fabien Chouteau - AdaCore, France

Alire (Ada LIbrary REpository) is a package manager project for the Ada/SPARK community. The goal of a package manager is to facilitate collaboration within the community and to lower the barrier of entry for beginners. In this talk we will present the Alire project, what it can do for you and how you can contribute and give more visibility to your Ada/SPARK projects. We will also provide a tutorial to show how to use Alire to create a library and then publish it for others to use.

17:00-17:20 - Protect Sensitive Data with Ada Keystore by Stephane Carrez - Twinlife, France

Storing passwords and secret configuration is a challenge for an application. Ada Keystore is a library that stores arbitrary content by encrypting them in secure keystore (AES-256, HMAC-256). The talk presents the project and shows how to use the Ada Keystore library to get or store secret information in a secure manner. The presentation explains how the Ada features such as types, protected types, tasks, pre/post conditions have helped during the development of this project.

17:30-17:50 - EUgen: a European Project Proposal Generator by Riccardo Bernardini - University of Udine, Italy

Whoever wrote a research project proposal knows how much unnerving it can be. The actual project description (made of work packages, tasks, deliverable items…) has lots of redundancies and cross-references that makes its coherency as frail as a house of cards. For example, if the duration of a task is changed most probably you'll need to update the effort in person-months of the task and of the including work package; you must update the start date of depending tasks and the delivery date of any deliverable items; most probably also the WP efforts and length need update too; not to mention the need of updating all the summary tables (summary of efforts, deliverable, ..) and the GANTT too. Any small changes is likely to start a ripple of updates and the probability of forgetting something and getting an incoherent project description is large. Given the harsh competition in project funding, if your project is incoherent the probability of getting funded is nil.

One day I got sick of this state of affair and I wrote my own project generator: 10k lines of Ada code that reads a non-redundant project description from a simple-format text file and produces a set of files ready to be imported in the proposal, GANNT chart included. The user can specify dependences between different items (e.g., this deliverable is produced at the end of this task, this milestone is reached when this deliverable is available, this task must begin after this other task...) and the program automatically computes all the dates. Both input parser and output processors are implemented using a plugin structure that makes it easy to write new parsers to read different formats or new output processors to produce output in different formats. Currently a parser for a simple ad-hoc format and an output processor that produces LaTeX files are provided; a new processor based on the template expander *protypo* is currently being implemented. Did I eat my own dog food? Well, yes, I did. I used it to write a proposal (still under evaluation) and it served me well.

18:00-18:20 - On Rapid Application Development in Ada by Tomasz Maluszycki - Poland

In the Ada world we typically write mission critical software that just has to work, but in a way one could argue that a lot more software is mission critical than is usually admitted. What does it take to actually perform rapid application development in any language? Can we do it in Ada and why would we do so? A quick look into some language features that can be [ab]used for enabling quick development of 'just a prototype' - which, as practice shows is often deployed into production, usually without proper quality controls and predictable outcome.

18:30-18:50 - Ada-TOML: a TOML Parser for Ada by Pierre-Marie de Rodat AdaCore, France

The world of generic structured data formats is full of contenders: the mighty XML, the swift JSON, the awesome YAML… Alas, there is no silver bullet: XML is very verbose, JSON is not convenient for humans to write, YAML is known to be hard to parse, and so on. TOML is yet another format whose goal is to be a good configuration language: obvious semantics, convenient to write and easy to parse in general-purpose programming languages. In this talk, I'll shortly describe the TOML format and show a few use cases in the real world. I'll then present the ada-toml library itself: its high-level architecture and examples.

18:50-19:00 - Informal Discussions & Closing

Informal discussion on ideas and proposals for future events.

More information on Ada Developer Room

Speakers' bios, pointers to relevant information, links to corresponding FOSDEM pages, etc., are available on the Ada-Belgium site at

www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem.html

We invite you to attend some or all of the presentations: they will be given in English. Everybody interested can attend FOSDEM 2020; no registration is necessary.

We hope to see many of you there!

Dirk Craeynest, FOSDEM Ada DevRoom coordinator

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/Ada-Europe/SIGAda/WG9)

Livestream for Ada Developer Room at FOSDEM 2020

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Subject: Livestream for Ada Developer Room at FOSDEM 2020
Date: Sat, 1 Feb 2020 09:17:27 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc, be.comp.programming

---

10th Ada Developer Room at FOSDEM 2020

Ada at the Free and Open source Software Developers' European Meeting
on Saturday 1 February 2020

Université Libre de Bruxelles (ULB), Solbosch Campus, Room AW1.125

Avenue Franklin D. Roosevelt Laan 50, B-1050 Brussels, Belgium

Organized in cooperation with Ada-Europe

www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem.html

fosdem.org/2020/schedule/track/ada

Today, February 1 2020, marks the start of the 20th edition of FOSDEM, the Free and Open source Software Developers' European Meeting, held this weekend in Brussels, Belgium.

In this edition, Ada-Belgium organizes once more a series of presentations related to the Ada Programming Language and Free or Open Software in a s.c. Developer Room. The "Ada DevRoom" at FOSDEM 2020 is held today, starting at 10:30. This year the Ada DevRoom has a total of 13 Ada-related presentations by 12 authors from 8 countries! There are also 4 more Ada-related presentations in other DevRooms.

If (like me) you can't be in the Ada DevRoom, follow the livestream or watch the recordings later!

On the Ada DevRoom page of the Ada-Belgium site, you see the schedule for the day, both in that DevRoom and others. Each entry points to the resp. page on the FOSDEM site, which has at the bottom the link for the livestream from the resp. room.

For the Ada DevRoom the live video stream is at:

https://live.fosdem.org/watch/aw1125

Enjoy!

Dirk Craeynest, FOSDEM Ada DevRoom coordinator

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/Ada-Europe/SIGAda/WG9)

FOSDEM 2020 Presentations & Videos

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Subject: FOSDEM 2020 Ada Developer Room - presentations & videos online
Date: Tue, 4 Feb 2020 14:19:28 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

*** Presentations and video recordings available online ***

10th Ada Developer Room at FOSDEM 2020

Saturday 1 February 2020

Brussels, Belgium

www.cs.kuleuven.be/~dirk/ada-belgium/events/20/200201-fosdem.html

fosdem.org/2020/schedule/track/ada

All presentations and video recordings from the 10th Ada Developer Room, held at FOSDEM 2020 in Brussels recently, are available via the Ada-Belgium and FOSDEM web sites now.

- "Welcome to the Ada DevRoom" by Dirk Craeynest - Ada-Belgium, Jean-Pierre Rosen - Ada-France

- "An Introduction to Ada for Beginning and Experienced Programmers" by Jean-Pierre Rosen - Adalog, France

- "HAC: the Compiler which will Never Become Big" by Gautier de Montmollin - Ada-Switzerland

- "Tracking Performance of a Big Application from Dev to Ops" by Philippe Waroquiers - Eurocontrol, Belgium

- "Cappulada: What we've Learned" by Johannes Kliemann - Componolit, Germany

- "Programming ROS2 Robots with RCLAda" by Alejandro R. Mosteo - Centro Universit. de la Defensa, Spain

- "Live Demo of Ada's Distribution Features" by Jean-Pierre Rosen - Adalog, France

- "Writing Shared Memory Parallel Programs in Ada" by Jan Verschelde - Univ. of Illinois at Chicago, USA

- "Spunky: a Genode Kernel in Ada/SPARK" by Martin Stein - Genode Labs, Germany

- "Alire: Ada Has a Package Manager" by Alejandro R. Mosteo - Centro Universit. de la Defensa, Spain & Pierre-Marie de Rodat and Fabien Chouteau - AdaCore, France

- "Protect Sensitive Data with Ada Keystore" by Stephane Carrez - Twinlife, France

- "EUgen: a European Project Proposal Generator" by Riccardo Bernardini - University of Udine, Italy

- "On Rapid Application Development in Ada" by Tomasz Maluszycki - Poland

- "Ada-TOML: a TOML Parser for Ada" by Pierre-Marie de Rodat - AdaCore, France

- "Securing Existing Software using Formally Verified Libraries" by Tobias Reiher - Componolit (in Security room)

- "BSP Generator for 3000+ ARM Microcontrollers" by Fabien Chouteau - AdaCore (in Hardware room)

- "Gneiss: A Nice Component Framework in SPARK" by Johannes Kliemann - Componolit (in Microkernels room)

- "A Component-based Environment for Android Apps" by Alexander Senier - Componolit (in Microkernels room)

Presentation abstracts, speaker bios, pointers to relevant information, copies of slides, links to corresponding pages and video recordings, are available via the Ada-Belgium and FOSDEM sites at the URLs above.

Some pictures will be posted later as well. If you have pictures or other material you would like to share, or know someone who does, then please contact me.

Finally, thanks once more to all presenters and helpers for their work and collaboration, thanks to all the FOSDEM organizers and volunteers, thanks to the many participants for their interest, and thanks to everyone for another nice experience!

Dirk Craeynest, FOSDEM Ada DevRoom coordinator

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/Ada-Europe/SIGAda/WG9)

#AdaFOSDEM #AdaDevRoom #AdaProgramming

#AdaBelgium #AdaEurope #FOSDEM2020

Make with Ada Winners Announced

From: dirk@orka.cs.kuleuven.be. (Dirk Craeynest)
Subject: Make with Ada 2019-2020 competition - winners announced
Date: Wed, 4 Mar 2020 18:49:43 -0000
Newsgroups: comp.lang.ada

I didn't see this mentioned on comp.lang.ada yet, so...

The winners of the latest "Make with Ada" programming competition [1] have been announced [2]: 10 winners were awarded a Finalist Prize, one of which got an additional First Prize and another one a Student Prize.

Congratulations to all winners!

[1] http://www.makewithada.org/

[2] https://www.hackster.io/contests/adacore2

Check out the many exciting projects!

Dirk

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/Ada-Europe/SIGAda/WG9)

*** 25th Ada-Europe Int'l. Conf. on Reliable Software Technologies ***

June 8-12, 2020 * Santander, Spain * www.ada-europe.org/conference2020

Ada-related Resources

[Delta counts are from Dec 2nd to Apr 2nd. —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: Thu, 2 Apr 2020 14:56:21 +0100
To: Ada User Journal readership

Ada groups on various social media:

- LinkedIn: 2_903 (+7) members [1]
- Reddit: 3_341 (+921) members [2]
- StackOverflow: 1795 (+49) questions [3]
- Freenode: 95 (+10) users [4]
- Gitter: 51 (+7) people [5]
- Telegram: 61 (+11) users [6]
- Twitter: 88 (+15) tweeters [7]
  169 (+80) 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: Thu, 2 Apr 2020 14:56:21 +0100
To: Ada User Journal readership

GitHub: 576 (+3) developers [1]

Rosetta Code: 707 (+41) examples [2]
38 (+2) developers [3]

Sourceforge: 271 (+1) projects [4]

Open Hub: 211 (+2) projects [5]

Bitbucket: 88 (+1) repositories [6]

Codelabs: 49 (+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: Thu, 2 Apr 2020 14:56:21 +0100
To: Ada User Journal readership

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

- TIOBE Index: 37 (+1) 0.23% (=) [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

Ada Learning Resources

[Follow-up to topic “Exercism” in AUJ 2019.4. —arm]

From: mario.blunk.gplus@gmail.com
Subject: Re: Ada learning resources
Date: Wed, 18 Dec 2019 10:47:59 -0800
Newsgroups: comp.lang.ada

Over the years I have put together lots of simple demo programs. The collection is growing. Perhaps it helps to understand Ada step by step. Your feedback is highly welcome.

https://github.com/Blunk-electronic/ada_training

From: charlet@adacore.com
Date: Sun, 15 Dec 2019 01:34:37 -0800

> There are a variety of Ada learning resources collected at https://www.adaic.org/learn/materials/, in a variety of forms (books, > tutorials, wikis, etc.).

By the way Randy,

https://learn.adacore.com/ should have a more prominent place in this page, it is the most up to date and well maintained training material for Ada, and is really much more than a tutorial, it contains complete training courses. It should be the first link IMO, instead of the current first section which is now getting outdated ("This series of articles is an introduction to Ada 95. The content is in the process of being updated to reflect the revisions introduced in Ada 2005 and the revisions currently underway for Ada 2012. But this is still an excellent introduction into the core technical features and benefits of Ada." isn't really the best advocate for recent training Ada material).

Ada-related Tools

Gnu Emacs Ada Mode Beta Test 7.0.0

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: Gnu Emacs Ada mode beta test 7.0.0
Date: Fri, 20 Dec 2019 09:15:42 -0800
Newsgroups: comp.lang.ada

Gnu Emacs Ada mode beta test 7.0.0 is now available at http://www.nongnu.org/ada-mode/. This is a significant refactoring, which may affect some user custom code, so it is not in Gnu ELPA yet.

To install, download the candidate ELPA archive, set package-archives to point to it, and use list-packages (more detailed instructions at http://www.nongnu.org/ada-mode/).

The wisi package now provides a more complete integration with Emacs project.el.

Several bugs have been fixed.

You may want to work thru the tutorials in ada-mode.info again; they now cover many of the new features.

See the NEWS files in ~/.emacs.d/elpa/ada-mode-7.0.0 and wisi-3.0.0, for more details. See wisi.info in the release for more information on the package.el integration.

Gnu Emacs Ada Mode 7.0.1

From: Stephen Leake <stephen_leake@stephe-leake.org>
Subject: Gnu Emacs Ada mode 7.0.1 released.
Date: Fri, 31 Jan 2020 06:01:11 -0800
Newsgroups: comp.lang.ada

Gnu Emacs Ada mode 7.0.1 is now available in GNU ELPA.

Relative to the previous Ada mode release (6.2.1), this is a significant refactoring, which may affect some user custom code.

The wisi package now provides a more complete integration with Emacs project.el.

You may want to work thru the tutorials in ada-mode.info again; they now cover many of the new features.

Relative to the previous beta test (7.0.0), this is a minor feature and bug fix release.

See the NEWS files in ~/.emacs.d/elpa/ada-mode-7.0.1 and wisi-3.0.1, 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-7.0.1
./build.sh
./install.sh
```

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

Qt5Ada 5.14.0 Free Edition

From: leonid.dulman@gmail.com
Subject: Announce: Qt5Ada version 5.14.0 (571 packages) release 13/12/2019 free edition
Date: Tue, 17 Dec 2019 06:34:49 -0800
Newsgroups: comp.lang.ada

Qt5Ada is Ada-2012 port to Qt5 framework (based on Qt 5.14.0 final) Qt5ada version 5.14.0 open source and qt5c.dll,libqt5c.so(x64) built with Microsoft Visual Studio 2019 in Windows, gcc x86-64 in Linux.

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

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

Changes for new Qt5Ada release:

Added new packages:

Qt.QTest for simulate mouse and keyboard events

Speech recognitions based on CMU Phenix

Prebuilt unofficial Qt 5.14.0 and VTK 8.2.0 win64 on Windows and x86-64 on *nix

My configuration script to build Qt 5.14.0 is:

configure -opensource -release -nomake tests -opengl dynamic -qt-zlib -qt-libpng -qt-libjpeg -openssl-linked

OPENSSL_LIBS="-lssleay32 -llibeay32" -plugin-sql-mysql -plugin-sql-odbc -plugin-sql-oci -icu -prefix "e:/Qt/5.14"

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 other things.

Qt5Ada and VTKAda for Windows, Linux (Unix)
https://r3fowwcolhrzycn2yzlzzw-on.drv.tw/AdaStudio/adastudio.html

Google drive:

https://drive.google.com/folderview?id=0B2QuZLoe-yiPbmNQRl83M1dTRVE&usp=sharing

(It can be mounted as virtual drive or directory or viewed with Web Browser)

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

VTKAda version 8.2 is based on VTK 8.2.0 (OpenGL2) is fully compatible with Qt5Ada 5.14.0

I hope Qt5Ada and VTKAda will be useful for students, engineers, scientists and enthusiasts. With Qt5Ada you can build any application and solve any problems easily and quickly.

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

VisualAda 1.2.5

From: alby.gamper@gmail.com
Subject: ANN: VisualAda (Ada Integration for Visual Studio 2017 & 2019) release 1.2.5
Date: Fri, 10 Jan 2020 14:05:44 -0800
Newsgroups: comp.lang.ada

Dear Ada Community,

VisualAda version 1.2.5 has been released.

Fixes include the following:

- Source code navigation implemented (ie goto definition and goto implementation).
- Quickinfo support has been added.
- Rudimentary statement completion support has been added.
- Project templates are now tagged appropriately under Visual Studio 2019, making it easier to find Ada related templates.

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

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

VisualAda 1.2.7

From: alby.gamper@gmail.com
Subject: ANN: VisualAda (Ada Integration for Visual Studio 2017 & 2019) release 1.2.7
Date: Sat, 8 Feb 2020 22:51:12 -0800
Newsgroups: comp.lang.ada

Dear Ada Community,

VisualAda version 1.2.7 has been released.

Enhancements include the following:

- Improved project load time (only load projects once if they are referenced multiple times within a solution)
- Improved statement completion response times (editor was significantly lagging when opening large projects / solutions)

Please feel free to download the free plugin from the following URL:
https://marketplace.visualstudio.com/items?itemName=AlexGamper.VisualAda

VisualAda 1.3

From: alby.gamper@gmail.com
Subject: ANN: VisualAda (Ada Integration for Visual Studio 2017 & 2019) release 1.3
Date: Fri, 17 Apr 2020 21:48:30 -0700
Newsgroups: comp.lang.ada

Dear Ada Community,

VisualAda version 1.3 has been released.

Enhancements include the following:

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

The runtime that is to be used must be selected in the "Ada RTS" property located in the "General" property page for the project.

Please feel free to download the free plugin from the following URL:
https://marketplace.visualstudio.com/items?itemName=AlexGamper.VisualAda

SDLAda, LÖVE, and Programming for Beginners

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject: sdlada, löve and programming for beginners
Date: Sat, 08 Feb 2020 12:40:54 +0100
Newsgroups: comp.lang.ada

At FOSDEM, my colleague Thomas Maluszycki gave a talk [1] about rapid application development in Ada. This made me think. You see, I have a 14-year-old son whom I teach programming to. He is lukewarm about it but I think it is my duty as a parent to give him basic education in this field, as computers are already everywhere and will probably govern his live even more than ours. So I played with him with Colobot[2], taught him a little bit of Ada (with the French translation of Barnes' book for Ada 95), a little bit of ZX Spectrum BASIC, and now he's writing a Pong clone with the LÖVE framework[3], in Lua[4]. This framework makes it very easy to have immediate results... but Lua lacks strong typing and in particular range checking, and a debugger.

So it occurred to me that LÖVE is really a Lua binding to SDL plus a predefined event loop, and that it would be quite easy to do something similar based on the sdlada thick binding. The goal would be to attract teenage programmers to the language and to programming in general. Possibly on a Raspberry Pi. I'd be willing to make a Debian package for it. What do you think?

[1] https://fosdem.org/2020/schedule/event/ada_rad/

[2] http://colobot.info/

[3] http://love2d.org/

[4] https://www.lua.org/

From: Lucretia <laguest9000@googlemail.com>
Date: Mon, 10 Feb 2020 06:27:31 -0800

On Saturday, 8 February 2020 11:41:01 UTC, Ludovic Brenta wrote:

Dragging this thread back on track...

> So it occurred to me that LÖVE is really a Lua binding to SDL plus a

I never looked at it before, but knew of it, never knew it was a wrapper around SDL.

> predefined event loop, and that it would be quite easy to do something similar based on the sdlada thick binding. The goal would be to attract

Yeah, that would be pretty cool. Any features required, just add a PR.

I want to get iterators around Surfaces (old, not really for new projects) and textures (for accelerated 2D and for new stuff).

Definitely not having to mess about with OpenGL/Vulkan is a good start.

> teenage programmers to the language and to programming in general. Possibly on a Raspberry Pi. I'd be willing to make a Debian package for it. What do you think?

Sounds good to me.

From: Chris Sykes <chris@amtiskaw.net>
Date: Tue, 11 Feb 2020 19:10:46 +0000

> So it occurred to me that LÖVE is really a Lua binding to SDL plus a predefined event loop, and that it would be quite easy to do something similar based on the sdlada thick binding. The goal would be to attract teenage programmers to the language and to programming in general. Possibly on a

> Raspberry Pi. I'd be willing to make a Debian package for it. What do you think?

FWIW, I think it's an excellent idea.

One of the most important things for a beginner is being able to achieve visible results from simple code. So something that allows you to draw to the screen and respond to user input, while minimum boiler-plate code (often confusing to newbies) really helps.

If you're looking for inspiration for some demos/examples, you should checkout the "One Lone Coder" videos on YouTube:

https://www.youtube.com/channel/UC-yuWVUplUJZvieEligKBkA/featured

He has written a really simple "game engine" in C++ along the same lines, and (IMO) his projects show just how valuable lowering the barriers to experimentation can be. Lots of fun too!

From: Lucretia <laguest9000@googlemail.com>
Date: Tue, 11 Feb 2020 11:25:40 -0800

> If you're looking for inspiration for some demos/examples, you should checkout the "One Lone Coder" videos on YouTube:

> https://www.youtube.com/channel/UC-yuWVUplUJZvieEligKBkA/featured

Can confirm OLC is very good / accessible, check out his SNES emulator series.

From: Rick Newbie <nuttin@nuttn.nowhere>
Date: Sun, 9 Feb 2020 09:34:16 -0800

[Starting here the thread goes on a tangent about Ada books and language complexity. —arm]

> [Original message quoted verbatim omitted. —arm]

[...]

On the topic of teenage programmers: Although I am not a teenager I am new to Ada. What is repelling is when you read Barne's book and you throw up your arms and think: How am I ever going to master all that?!

But I guess what is true for C++ must be true for Ada as well: People use 20% of the language features 80% of the time. it would be good to find a way to introduce new programmers using these 20% to start with. Barne's book is simply overwhelming for the newcomer since it covers nearly all aspects and you can start out with much less

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sun, 09 Feb 2020 23:55:05 +0100

> On the topic of teenage programmers: Although I am not a teenager I am new to Ada. What is repelling is when you read Barne's book and you throw up your arms and think: How am I ever going to master all that?!

Even though it is out of date by now, I still like and recommend the free book by John English, "Ada 95: the Craft of Object-Oriented Programming". This is a gentle introduction to Ada as a first programming language and it is not overwhelming. Professionals and die-hard enthusiasts can always learn from the reference manual

https://www.adaic.org/resources/add_content/docs/craft/html/contents.htm

From: "Nasser M. Abbasi" <nma@12000.org>
Date: Sun, 9 Feb 2020 22:53:13 -0600

> On the topic of teenage programmers: Although I am not a teenager I am new to Ada. What is repelling is when you read Barne's book and you throw up your arms and think: How am I ever going to master all that?!

One of the reasons a programming language become less popular is that it becomes more complicated with time.

Look at what happened to C++. Same with Ada. They start relatively small and simple, and each few years, they update the standard and add more complication and "advanced" features so that few could understand it all. This has also happened to Fortran with addition of OO to it, where it is as complex as C++ and Ada. Fortran used to be a very simple language.

One of the reasons why python is so popular (even though I think it is a horrible language myself) is that it is "simple".

There should be something in between. A simple, yet well designed and strongly typed language. That is why I liked Pascal the most of all the languages I programmed in (followed by Ada).

From: Rick Newbie <nuttin@nuttn.nowhere>
Date: Mon, 10 Feb 2020 02:05:45 -0800

That's actually very true. I have to work in C++ professionally but I always remember the day of Turbo Pascal or Modula-2. Install, run the IDE and ready to go, no fighting about missing libraries or esoteric features. I must admit that I look at some new C++ programs and I don't understand what's going on. Same with forums. Sometimes I browse Stackexchange just for fun and I read questions from people about the behavior of pieces of code that they don't understand and the answers just make me shake my head. Who would have ever thought of that?!

When I learned C I had a book about 200 pages. I read that and afterwards I was able to write my first small programs. I don't have the feeling it will be that easy with Ada. In fact I try to keep it simple and get me some exercise by translating some of the games from David Ahl's 1970's book from BASIC to Ada because I think that it is possible to translate those games with the more simple features of Ada to get me going.

I think when the language becomes so complicated that you need professional help, not with algorithmic problems but with syntactical questions, it is too bloated. Hence my above remark that you use 20% of the features 80% of the time. I know certain modern features are a blessing, for instance I love Lambdas in C++ because they allow me to put active code in a datatable instead of in a long switch statement, but I could live without it if necessary.

If I remember my early teachings correctly you can formulate nearly every problem on a Turing Machine.

Stack Usage

From: Simon Wright <simon@pushface.org>
Subject: ANN: Stack usage
Date: Thu, 20 Feb 2020 22:24:02 +0000
Newsgroups: comp.lang.ada

Want a worst-case estimate of your embedded app's stack usage? (so you can allocate your tasks only enough stack to avoid stack overflow ...)

https://github.com/simonjwright/stack_usage

[Summary from the above link follows. —arm]

The Python program stack_usage.py is intended to help with this (it's not a panacea, though! if you have AdaCore support, you'll be better off using GNATstack).

The initial motivation for this work was a hard fault encountered while writing a test program to check that Ada timing events work properly (well, usably) with the FreeRTOS-based Cortex GNAT RTS.

stack_usage has been developed on macOS Mojave using Python 2.7 and 3.7 and PLY (Python Lex and Yacc). To install PLY,

```
pip install --user ply
```

It relies on the information generated by the GCC compiler (FSF GCC 10 or later, GNAT GPL 2015 or later) using the switch -fcallgraph-info=su,da.

Threefish-256

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Subject: Threefish-256
Date: Sun, 16 Feb 2020 17:46:34 +0100
Newsgroups: comp.lang.ada

I have made an implementation of the Threefish-256 encryption algorithm available at https://github.com/jrcarter/Threefish in the hope that some will find it useful.

While I think it's correct, I cannot be sure, and would appreciate additional people inspecting it for errors.

KDF9 Emulator Release 4.1d

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Subject: ee9, KDF9 emulator release 4.1d
Date: Wed, 26 Feb 2020 18:13:50 +0000
Newsgroups: comp.lang.ada

If you are interested in historical computers you might like to take a look at the new release of ee9, my emulator of the English Electric KDF9, now available here:

http://www.findlayw.plus.com/KDF9/#Emulator

where you can find pre-built binaries for macOS and Linux. I expect that the Linux binary will also work on Windows 10, but have not tried that.

For the first time, this release of ee9 enables both the use of the Kidsgrove optimising Algol 60 compiler and the execution of object programs under the control of the Time Sharing Director, KDF9's elegantly simple multiprogramming operating system.

In the download package are papers describing the hardware and software of the KDF9, and its role in the development of machine-independent benchmarking.

ee9 and its ancillary programs are written in Ada 2012 (of course). Source code, and instructions on using the build process, are also included.

From: Paul Rubin <no.email@nospam.invalid>
Date: Wed, 26 Feb 2020 12:49:38 -0800

Cool! Do you have the KDF9 source code for the Algol 60 compiler or the timesharing OS?

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Thu, 27 Feb 2020 00:47:48 +0000

The ee9 download includes listings of the Whetstone Algol system.

You can find a lot more original or resurrected KDF9 material, starting here:

http://sw.ccs.bcs.org/KDF9/index.html

See also the Bibliography included in the download.

Simple Components v4.47

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components for Ada v4.47
Date: Sun, 1 Mar 2020 13:11:28 +0100
Newsgroups: comp.lang.ada

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, streams, multiple connections server/client designing tools and protocols implementations. The library is kept conform to the Ada 95, Ada 2005, Ada 2012 language standards.

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

Changes to the previous version:

- MODBUS RTU client implementation was added;
- On_Reader_Start and On_Writer_Start primitive operation were added to Blocking_Server;
- On_Start primitive operation was added to Call_Service;
- On_Pooled_Server_Start primitive operation was added to Pooled_Server;
- On_Worker_Start primitive operation was added to Connections_Server;
- Activated primitive operation was added to Connection.

Zip-Ada v56

From: gautier_niouzes@hotmail.com
Subject: Ann: Zip-Ada v.56
Date: Thu, 26 Mar 2020 10:02:16 -0700
Newsgroups: comp.lang.ada

New in v.56:

- Zip: the Zip_info type is now controlled (no need to call Delete; additionally, clones are done correctly).
- UnZip.Streams: added Size and Name functions for Zipped_File_Type.
- LZ77: added nice simple LZ77 compressor by Rich Geldreich, Jr.
- (Tools) Added Zip_Dir_List.

New in v.55:

- Zip_Streams: ZS_Size_Type is now 64-bit signed, enabling Zip.Create to capture archive size overflows in Zip_32 mode.
- Zip.Create raises Zip_Capacity_Exceeded when archive creation exceeds the Zip_32 format's capacity: 4GB total size, 65,535 entries.
- Zip.Create is now using an Ada 2005+'s Containers's Hashed Maps; creation is much faster on Zip archives with many entries.
- (Tools) ReZip has a new option for working only with its own internal compression algorithms - those provided by Zip.Compress. This option is useful if external tools are not available.
- New Trained_Compression package: generic streaming encoder-decoder engine with the capability of training the engine with data known in advance, in order to achieve better compression. Not Zip-related.

-!- Minimum required Ada version is now Ada 2005 (was Ada 95 before).

Full history: http://unzip-ada.sf.net/hist.htm

Main site & contact info:

http://unzip-ada.sf.net

Project site:

https://sf.net/projects/unzip-ada/

GitHub clone:

https://github.com/zertovitch/zip-ada

AdaNetFramework - Proof of Concept Alpha Release

From: alby.gamper@gmail.com
Subject: Ann: AdaNetframework - Proof of concept / alpha release
Date: Thu, 26 Mar 2020 02:11:35 -0700
Newsgroups: comp.lang.ada

Dear Ada Community

For those interested in Microsoft NetFramework, I have developed a set of bindings, runtime that allows native Ada applications built using GNAT to use the NetFramework. Conceptually this is very similar to "Embedinator 4000" developed by the "Mono" development team. Note this is not Ada compiled into CLR/VM bytecode, but a native (albeit for the moment) Windows x64 application that can make use of the functionality provided by the NetFramework.

Note this is a Proof of concept/alpha release, but it is functional

The git repo contains 3 branches, these being

1) Master - a cutdown version of mscorlib (only includes subset of mscorlib)
2) System.dll - contains the core system bindings (core dependency)
3) System.Windows.forms.dll - contains winforms bindings

I suggest that if you want to build/test the repo, please start with the "Master" branch (which contains a rudimentary test application (i.e., VS/GPR project) and then progress to System and finally System.Windows.Forms branch. Note that the Winforms branch will take ~45 min to complete, so be patient (it's a large lib!)

Notes:

1) Please use the latest version of VisualAda to build the projects. There was a memory leak which may/will cause the final part of the build to fail
2) I am intending to support NetCore going forward, so that Mac, Linux clients will be supported. But this may

take some time, since the CLR hosting /interop Api's are very different from NetCore to NetFRamework

Git repo is https://github.com/Alex-Gamper/Ada-NetFramework.git

Feel free to raise questions / comments

From: Shark8 <onewingedshark@gmail.com>
Date: Tue, 31 Mar 2020 08:25:18 -0700

So, this allows you to use DOTNET stuff in native applications; that's pretty nifty.

There was a GANT that targeted DOTNET directly, though the latest one is pretty old now (2014?) and there was an Ada spec generator where you could import DOTNET libraries for use in your Ada programs. [I have a copy of Delphi.NET, so of course I ran it over the Delphi DLLs. ;) And had some fun playing around with that.]

From: alby.gamper@gmail.com
Date: Fri, 3 Apr 2020 23:27:39 -0700

Hi Shark8

Yes it does allow you to use DOTNET directly (ie from a native x64 binary) The big difference between my approach and targeting CLR/DOTNET directly as the BNAT 2014 did, is that the Bindings need to be generated based on a predefined set of assemblies. Hence the reason for the 2 main branches in Git

The "system.dll" branch contains only the bindings for system.dll and the "System.windows.Forms.dll" branch contains the bindings for WinForms and all its dependencies (If there is demand I can do another branch for WPF)

Ada and Operating Systems

Ada and macOS Cocoa

From: Matt Borchers <mattborchers@gmail.com>
Subject: Ada development resources for Mac OSX Cocoa applications
Date: Sun, 8 Mar 2020 17:10:58 -0700
Newsgroups: comp.lang.ada

Doing a search here for "mac osx cocoa" returns one hit from 1999. Needless to say, any "help" in that thread would be woefully outdated.

I am new to a team that needs to port a 32-bit Carbon graphical desktop application written in Ada to its 64-bit Cocoa equivalent. Searching the web for any kind of Cocoa function library (in the same spirit as .NET) is leading me to very little of use aside from what I find at developer.apple.com. Apple developers seem to care mostly about iOS/phone development than desktop apps. Can anybody direct me to a good set of reference guides, on-line or off-line, that would be helpful to a programmer writing a graphical Mac application using non-native (i.e. non-Apple) tools. Being old-school, I would even appreciate the name of a good book.

I know that 20 years ago interfacing Ada to Objective-C was a very difficult task. I am hopeful that the past 20 years have brought technologies that have made this easier or even simple. Has anybody written an Ada binding to AppKit or Foundation with good documentation?

From: Optikos <ZUERCHER_Andreas@outlook.com>
Date: Tue, 10 Mar 2020 07:09:31 -0700

The Carbon-to-Cocoa in any language is going to be a near-total rewrite. While performing that rewrite, you might consider rewriting in a nonAda language (e.g., Swift) anyway.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Tue, 10 Mar 2020 18:36:44 +0100

> That isn't the reply I was hoping to read, but it is the one I expected.

Not what you're asking, but I would suggest you use Gnoga for your desktop applications (which Gnoga refers to as "singleton" applications). Then your code will be portable across platforms.

Ada Practice

Type Naming Conventions: Any_Foo

[As this topic involves a degree of personal preference, a large conversation sprung around this question, requiring a more extensive cherry-picking of posts. I have selected what I feel most relevant, but I apologize to the topic contributors if they see some of their answers omitted. —arm]

From: "Alejandro R. Mosteo" <alejandro@mosteo.com>
Subject: Type naming conventions: Any_Foo
Date: Wed, 4 Dec 2019 *14:56:21 +0100*
Newsgroups: comp.lang.ada

I've recently come across a new (to me) type naming convention and I'm curious about how extended it is. I was aware of the

```
Foo.Object -- where Foo is a package and
           -- Object is the type name
```

and

```
Foos.Foo -- where Foos is a package and
         -- Foo is the type
```

and

```
Foos.Bars -- where both packages and
          -- types are in plural
```

and

```
Foo_Type -- where the enclosing package
         -- name is not used
```

This variant is

```
Any_Foo -- enclosing package also not
        -- used
```

I've found only one example in the ARM in System.Any_Priority. I find I like better Any_Foo than Foo_Type, not sure why. I've had since I can remember an aversion for the _Type thing.

Anyway, just curious. Any champions of the Any_Foo in the readership?

From: "Alejandro R. Mosteo" <alejandro@mosteo.com>
Date: Wed, 4 Dec 2019 17:42:56 +0100

> Not come across this, is this for when "use" is used? What's the name of the package, Foos? Foo?

I was ambiguous, sorry. In this case I think the enclosing package is secondary. I guess the advantages are the same as in _Type, that you can write Foo: Any_Foo;

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Thu, 5 Dec 2019 02:51:34 -0800
Of all of these schemes, my favorite is

```
Package  Foos
Type     Any_Foo
Object   Foo
```

This is tightly related to the discussion predefined types vs. user defined types. It's not always easy, ahem it's often difficult to find good names.

I think finding good names and spending time on this is well spent effort.

I do not know who posted this example a long time ago, but I like it:

Do not use abbreviations. Good names make a program understandable. What is Wpn?

```
type Weapon_Type is (Broadsword,
     Catapult, Bow_and_Arrow);
procedure Attack_Using (Weapon:
     Weapon_Type);
Weapon: Weapon_Type;
Attack_Using (Weapon => Catapult);
-- a bit talkative
Attack_Using (Catapult);
-- good only with positional association
```

versus

```
type Weapon is (Broadsword, Catapult,
     Bow_and_Arrow);
procedure Attack (Using: Weapon);
My_Weapon, Foes_Weapon: Weapon;
Attack (Using => Catapult);
-- good only with named association
Attack (Catapult);
-- Do we attack the catapult or what?
```

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Fri, 6 Dec 2019 00:57:06 -0800

> procedure Attack (Using: Weapon);

> Attack (Using => Catapult);

As nice as this may read in user's code, within the body of Attack, the parameter

name is not optimal.

Also a declaration like

```
My_Weapon: Weapons.Weapon;
```

is awkward when use-clause is banned. So a further point to consider is whether you want your package to be used with use-clause or without:

```
My_Weapon: Weapons.Object;
```

I'm not sure I like this.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Fri, 6 Dec 2019 10:55:28 +0100

Small remark: do not confuse using the use clause, and not using selected names. You are perfectly allowed to use selected names within the scope of a use clause if you feel it is more readable!

I am a known supporter of the use clause, however for classes, I use the package for the object name, and "object" for the record that's the data part of it. Of course, I always use selected names in that case.

[small plug] There is an AdaControl rule to check that some names always use selected notation.

Whether or not you are hostile to the use clause, the best advice is to choose a name which is nice for your favorite notation, and acceptable for the other one.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Thu, 5 Dec 2019 18:27:24 +0100

> (Broadsword, Catapult, Bow_and_Arrow);

There are 2 ways to look at them:

1. These identify the possible weapons: Weapon_ID

2. These are the names of the possible weapons: Weapon_Name

Either of these are better than any name derived using a convention, while still leaving the best name (weapon) available for parameter names. This because they were created by thinking (which is what S/W engineers are paid to do), while conventions exist to allow developers to avoid thinking.

I would even say that those who use naming conventions such as T[y[p[e]]] are either not S/W engineers or are shirking their duties.

From: Lucretia <laguest9000@googlemail.com>
Date: Fri, 6 Dec 2019 03:44:46 -0800

[...]

An alternative for enumerations would be:

```
type All_Weapons is (Broadsword,
      Catapult, Bow_and_Arrow);
Weapon : All_Weapons := Broadsword;
```

But this does look a bit weird, maybe a renaming of All_Weapons to A_Weapon would make the variable definition look better?

```
Weapon : A_Weapon := Broadsword;
```

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Fri, 6 Dec 2019 21:23:45 +0100

It has been demonstrated that the first few characters of an identifier are the most important in distinguishing them; having lots of identifiers with the same first few characters makes the code harder to read. So common prefixes are even worse than suffixes.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Fri, 6 Dec 2019 23:55:59 +0200

Some people have suggested decorating the variable/component name instead of the type name, for example

```
function Is_Lethal (The_Weapon :
      Weapon) return Boolean
```

but (as you say) such prefixes are worse than suffixes.

I have also seen coding rules that require specific suffixes on formal parameters, such as:

```
function Is_Lethal (Weapon_P : Weapon)
      return Boolean;
```

but they tend to also require suffixes (like "_T") on type names, so there we are again.

From: "*Dmitry* A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 6 Dec 2019 21:46:33 +0100

>> But if a value of the type Weapon_Id is an identifier of a Weapon, how can you defend saying

>>

>> Weapon : Weapon_Id;

>>

>> The variable Weapon does not represent a Weapon; it represents an identifier of a Weapon, so the name Weapon is IMO a little misleading.

> Obviously there are no weapons in the S/W; there are only bit patterns that you have decided to interpret in various ways. But if you're modeling the problem space and it contains something called Weapon, then

> your software had better have something named Weapon it in, too.

Which something is the variable Weapon in the example above. Though Holstered_Weapon, Current_Weapon might be better. But then again, "_Weapon" would look like a nasty suffix.

I don't think there is a universal solution and I agree with Randy that a consistent convention is better than anything else (unless pushed ad absurdum like Hungarian notation).

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Thu, 5 Dec 2019 18:45:05 +0100

> I would even say that those who use naming conventions such as _T[y[p[e]]] are either not S/W engineers or are shirking their duties.

There exist cases:

1. Formal generic types. They are customarily named XXX_Type.

2. Types which are artifacts of language issues or of design. These have no separate problem space meaning and thus no meaningful name. E.g.

```
type Something is ...;
type Something_Ptr is access Something;
   -- I don't want access type,
   -- I am required to have it
```

BTW, this includes all sorts of helper types Ada kept introducing recently.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Thu, 5 Dec 2019 22:51:36 +0100

> On 12/5/19 6:45 PM, Dmitry A. Kazakov wrote:

>>

>> 1. Formal generic types. They are customarily named XXX_Type.

>

> Well chosen names for generic formal types do not end with _Type. The

> PragmAda Reusable Components have many generic formal types, none of

> which end with _Type.

Ada standard library uses _Type, e.g.

```
generic
   type Element_Type (<>) is private;
   with function "=" (Left, Right :
Element_Type) return Boolean is <>;
package
Ada.Containers.Indefinite_Holders
```

As I said, the rationale is that there is no meaningful name for Element_Type in the problem space. There is no problem space at all. Indefinite_Holders is a helper package so general that considered in isolation it has no meaning.

>> -- I don't want access type, I am required to have it

> Please provide examples of being required to have an access type.

There are lots of cases in Ada, you certainly should know that. As a

practical example GtkAda declares all widget types twice:

```
type Gtk_Button_Record is ...
type Gtk_Button is access all
      Gtk_Button_Record'Class;
```

The suffix _Record is an equivalent to _Type.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Thu, 5 Dec 2019 17:12:57 -0600

> There are lots of cases in Ada, you certainly should know that. As a practical example GtkAda declares all widget types twice: [...]

This is the only case where I've used "Any_" as a type prefix, in Claw -- specifically, class-wide access types.

```
type Root_Window_Type is abstract
      tagged private;
type Any_Window_Access_Type is
     access all Root_Window_Type'Class;
```

Access-to-classwide is a different sort of thing than access-to-specific, and I wanted a different sort of name for it.

[...]

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 6 Dec 2019 18:57:19 -0600

[The conversation veers towards examples in the ARM. —arm]

> On 2019-12-06 21:18, Jeffrey R. Carter wrote:

>> On 12/5/19 10:51 PM, Dmitry A. Kazakov wrote:

>>>

>>> Ada standard library uses _Type, e.g.

>>>

>>> generic

>>>   type Element_Type (<>) is private;

>>>   with function "=" (Left, Right : Element_Type) return Boolean is <>;

>>> package Ada.Containers.Indefinite_Holders

>>

>> Yes, and the ARM also includes such abominations as anonymous access types. Just because it's in the ARM doesn't mean it's the best way to do something. Element is be [sic] a better name for that formal type.

>

> No, it would be misleading. Element must be reserved for instances of the type. They are actual elements. The type of an element is not an element, these are two totally different things.

Agreed. Ada.Containers all have a function Element that retrieves a (copy of) a single element object from the container. If the type was named element, what would this function be called? Similarly, some of the parameters are called Element (thus, Element: Element_Type in many parameter lists); those also would need alternate names.

There were a number of ARG members that disliked the "_Type" notation, so we looked at alternatives. And we didn't find anything that worked as well. Sometimes, package design is about the "least bad" alternative.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 7 Dec 2019 14:36:51 +0200

>> Agreed. Ada.Containers all have a function Element that retrieves a (copy of) a single element object from the container. If the type was named element, what would this function be called? [...]

>

> At a conference long ago (probably in the Ada-83 days), a presenter claimed that well designed Ada has 90% of its operations named Put or Get.

Horror. The result of blind OO convention :-)

> Get is an appropriate name for such an operation.

I follow the convention that procedure names are verbs ("Get") or verb phrases ("Remove_Last_Item") that describe the action or its effects, while function names are nouns ("Element") or noun phrases ("Largest_Element") that describe the value returned by the function.

Therefore, IMO, "Get" is not a proper name for a function (unless the program models animal breeding, and the Get function returns all the offspring of a particular animal or pair of animals).

[...]

Importance of GNAT.Source_Info

From: Jere <jhb.chat@gmail.com>
Subject: Importance of GNAT.Source_Info
Date: Mon, 6 Jan 2020 14:03:54 -0800
Newsgroups: comp.lang.ada

I'm working on a baremetal RTS and while looking at https://wiki.osdev.org/Ada_Bare_bones, one of the files it suggests is part of the minimum set of RTS files is g-souinf.ads which contains the package GNAT.Source_Info.

Does anyone know what part of the compiler requires this? So far I haven't had GNAT barf at me for not having it while compiling the files I do have, but I don't want to leave it out if it is indeed necessary for something. I didn't see any info on why it is necessary. It's definitely useful in that it gives a lot of compile time values, but not sure why it would be a "required" file. I'm assuming something will break without it, but don't know what. [...]

From: charlet@adacore.com
Date: Fri, 10 Jan 2020 02:54:18 -0800

This package is completely optional and standalone, and not used by the compiler or other runtime units. It's just provided because it doesn't have any associated runtime so portable to all GNAT ports and because it's convenient. You can safely remove or ignore it if that's convenient for you.

Peculiarities of "of" Iteration Syntax

From: "Alejandro R. Mosteo" <alejandro@mosteo.com>
Subject: Peculiarities of "of" syntax
Date: Sat, 11 Jan 2020 19:05:21 +0100
Newsgroups: comp.lang.ada

[...]

The following GNAT 2019-rejected examples of iterating with the new "of" are giving me some pause if this is an oversight in the feature, a bug in the compiler, or actually intended for some good reason:

```
procedure Pro is
   type Int_Array is array (Positive range <>)
         of Integer;
   Arr : Int_Array (1 .. 10) := (others => 0);
begin
   -- 1)
   for Z of Arr loop -- of course valid
      null;
   end loop;

   -- 2)
   for Z of Int_Array'(Arr & Arr) loop
   -- also works
      null;
   end loop;

   -- INVALID
   -- 3)
   for Z of Arr & Arr loop
     -- Error is "missing loop"
     null;
   end loop;

   -- 4)
   for Z of (Arr & Arr) loop
      -- Error is "name expected"
      null;
   end loop;

end Pro;
```

The crux of the matter might be in 5.5.2, where an "iterator_name" appears:

iterator_specification ::= defining_identifier in [reverse] iterator_name | defining_identifier [: subtype_indication] of [reverse] iterable_name

http://www.ada-auth.org/standards/rm12_w_tc1/html/RM-5-5-2.html

[...]

From some other related question [1] I need to review the master rules in 7.6, but my first instinct is that 3) and 4) could be legal if the loop is the master of the expression.

In conclusion: any insight on what's going on with 3) and 4)? Thanks!

[1] https://groups.google.com/d/msg/comp.lang.ada/veW6BNBGBfo/gDFwEsyyAgAJ

From: Robert A Duff <bobduff@TheWorld.com>
Date: Sat, 11 Jan 2020 16:45:07 -0500

The syntax rules require a name after "of", and neither "Arr & Arr" nor "(Arr & Arr)" are names. [...]

From: "Alejandro R. Mosteo" <amosteo@unizar.es>
Date: Sat, 11 Jan 2020 17:38:00 -0800

So the question would be [...] why a name is required instead of an expression, or why a qualified expression is good enough but an unambiguous plain expression is not.

A normal function call will work there too, so why not the poor infix operator...

From: Simon Wright <simon@pushface.org>
Date: Sun, 12 Jan 2020 11:27:22 +0000

This is OK too:

```
-- 3a)
for Z of "&" (Arr, Arr) loop
  null;
end loop;
```

[...]

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Mon, 13 Jan 2020 17:32:36 -0600

[...]

A qualified expression, a type conversion, and a function call are all "names". You can always use a qualified expression to turn any "expression" into a "name".

> A normal function call will work there too, so why not the poor infix operator...

The usual reason is that infix operators would make the grammar ambiguous; that depends on what follows them. There are cases where that isn't a problem (this might be one of them). I don't think anyone was thinking about using expressions in this context, the intent was to iterate over an object.

Of course, Bob is right that the difference between "name" and "expression" (and similarly "value" and "object") are minor enough that it would be nice to eliminate them. (But it's also a lot of work, and we decided not to try for Ada 202x.)

Ada 202X Syntax Legibility Concerns

[This subthread derived from an unrelated question about counting occurrences of numbers in a sequence. —arm]

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Subject: Re: Tally
Date: Tue, 14 Jan 2020 22:08:07 +0100
Newsgroups: comp.lang.ada

> [...] I would like to get an advice on how to program in a simple and fast way the following [...]:
>
> Example_Input: (2, 3, 8, 2, 2, 2, 7, 2, 3, 4, 8) ; -- variable Length
>
> Output by function or procedure: ((2, 5), (3, 2), (8, 2), (7, 1), (4, 1)); -- unknown Length

A lot depends on what restraints the problem domain puts on the input values. If you can define something like

```
type Input_Number is range 1 .. 10;
```

then you can do something straightforward like

```
type Count_Map is array (Input_Number)
      of Natural;
Count  : Count_Map := (others => 0);
Number : Input_Number;
...
loop
   exit when No_More_Numbers;

   Number:= Next_Number;
   Count (Number):= Count (Number) + 1;
end loop;
```

If you can't limit the input numbers to a sufficiently small range that an object like Count can be declared, then you'll need to use a map, as Holsti suggested, which is only a little more complicated.

From: Brad Moore <bmoore.ada@gmail.com>
Date: Thu, 16 Jan 2020 07:35:28 -0800

In Ada 2020, I think one could use a container aggregate to initialize the map to contain the initial objects:

e.g.

```
package Count_Maps is new
   Containers.Ordered_Maps
    (Key_Type     => Natural,
     Element_Type => Natural);

Counts : Count_Maps.Map
  := [for I in Input'Range
      when (for all J in Input'First .. I - 1 =>
              Input (I) /= Input (J))
        use I => 0];
```

The "when" clause should filter the input to just the unique values.

The "use" clause creates the mapping between key value and count value (Initially 0).

Then you could just write:

```
for Number of Input loop
   Counts (Number):= Counts (Number) + 1;
end loop;
```

To update the counts.

I leave it to the reader to decide whether this is clearer than what you had, as I think many would prefer what you had.

For that matter, you could do it all in one shot and even make the map a constant.

```
Counts : constant Count_Maps.Map
  := [for I in Input'Range
      when (for all J in Input'First .. I - 1 =>
              Input (I) /= Input (J))
        use I =>
          [for K in Input'Range when Input
                (K) = Input (I)) => 1]
           'Reduce("+", 0)];
```

Using a reduction expression to count the values. But this is definitely quite a mouthful.

I think if one wanted a constant object, it would be clearer to write a function that returns the map container object using a simpler form of expression to create the return object.

It would be nice however, if one could test membership of an array or container using "in" or "not in" to see if a particular element value can be found.

Then one could write;

```
Counts : Count_Maps.Map :=
  [for I in Input'Range when (Input (I) not in
      Input (1 .. I - 1)) use I => 0];
```

To create the initial map objects, which is easier to read.

Similarly, it would be nice to apply 'Max or 'Min to an array or container object, which could be shorthand forms for reduction expressions using Ada 2020 syntax to return the largest or smallest element in the array or container.

e.g.

```
Put_Line ("Biggest=>" &
            Natural'Image(Input'Max));
```

But if these ideas have any merit, you'd have to look past Ada 2020 to a future version of the language.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Thu, 16 Jan 2020 14:20:13 -0600

> Counts : constant Count_Maps.Map
> := [for I in Input'Range
> when (for all J in Input'First .. I - 1 => Input (I) /= Input
> (J))
> use I =>
> [for K in Input'Range when Input (K) = Input (I)) => 1]
> 'Reduce("+", 0)];
>
> Using a reduction expression to count the values. But this is definitely quite a mouthful.

So Ada 202x will allow us to catch up to C++ and many other "expressive" languages by allowing us to have "guess what this code does" contests!! :-)

Compared to Jere's loop, the above is impenetrable. And it's hard to guess the

performance of that (I'd have to expand the aggregate into its underlying operations to figure out whether it is more or less expensive than the simple loop would be).

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Thu, 16 Jan 2020 23:00:58 +0100

> Counts : Count_Maps.Map

> := [for I in Input'Range

> when (for all J in Input'First .. I - 1 => Input (I) /= Input (J))

> use I => 0];

I think you want

use Input (I) => 0

here (and further on). I can figure out what this does, but I wouldn't call it clear.

> Counts : constant Count_Maps.Map

> := [for I in Input'Range

> when (for all J in Input'First .. I - 1 => Input (I) /= Input (J))

> use I =>

> [for K in Input'Range when Input (K) = Input (I)) => 1]

> 'Reduce("+", 0)];

This is getting close to write-only code.

From: Brad Moore <bmoore.ada@gmail.com>
Date: Thu, 16 Jan 2020 18:51:39 -0800

> > This is getting close to write-only code.

> Already there.

That'll be the challenge, I think. With more tools (and new ones) to work with, one hopes people will end up choosing the right tool for the job. Some of the new tools are powerful, and there might be a tendency to want to use them, but a simpler tool can get the job done faster sometimes.

This example feels like using a big new table saw to slice bread, when a good 'ol bread knife can get it done faster and better.

Note that the simple loop accomplishes the task with a single pass through the date. The monster expression has 3 levels of nested loops, so hard to imagine it would beat the simple loop.

Getter Functions vs Record Components

[The thread discusses visibility of record components taking precedence over functions with the same name when using dot notation. —arm]

From: reinert <reinkor@gmail.com>
Subject: Is this a bug?
Date: Mon, 30 Dec 2019 07:44:35 -0800
Newsgroups: comp.lang.ada

Hello,

assume the following Ada procedure:

```
-----------------------------------------------------

with Text_IO;
procedure test1 is

   package test_package is
      type rec1_t is tagged record
         a : integer := 2;
         -- b : integer := 2;
      end record;
      function a(x : rec1_t) return integer
                  is (3);
         rec1 : rec1_t;
   end test_package;

begin
   Text_IO.Put(" test_package.rec1: " &
         Integer'image(test_package.rec1.a));
end test1;

-----------------------------------------------------
```

It gives (for my computer):

```
test_package.rec1: 2
```

If I change the statement

```
"a : integer := 2;"
```

to

```
"b : integer := 2;"
```

then I get:

```
test_package.rec1: 3
```

Is this reasonable? Bug?

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Mon, 30 Dec 2019 20:41:07 +0200

When rec1_t is tagged, the "selected component" text "test_package.rec1.a" could refer either to the rec1_t-component "a" or to the subprogram (function) "a". In RM 4.1.3(9.1/2) and RM 4.1.3(9.2/3), the latter case is available only under the condition that the tagged record type (rec1_t) does not have a (visible) component with the name "a". This means that the ambiguity is resolved in favour of the component "a", which has the value 2.

One could ask, why is such an ambiguity not rejected (made illegal)? Probably because such an illegality rule would have made illegal many Ada programs that were legal before the introduction of the "object.operation" syntax for tagged-record objects.

If this is a problem for you, you might check if your compiler has an option to warn about such cases, or if AdaControl can do the same.

From: "J-P. Rosen" <rosen@adalog.fr>
Date: Tue, 31 Dec 2019 07:08:39 +0100

> If this is a problem for you, you might check if your compiler has an option to warn about such cases, or if AdaControl can do the same.

Not yet, but it's a good idea. I keep it as an improvement suggestion.

From: reinert <reinkor@gmail.com>
Date: Mon, 30 Dec 2019 11:50:37 -0800

> One could ask, why is such an ambiguity not rejected (made illegal)? Probably because such an illegality rule would have made many illegal many Ada programs that were legal before the introduction of the

"object.operation" syntax for tagged-record objects.

I have had the understanding that the *intention* of primitive operations of tagged (record) types in some way can be looked at as an extension of the actual record - especially if one uses the dot notation. In this case I would expect (at least) a warning from the compiler.

I discovered the ambiguity when I accidentally did put in an extra component in a tagged record and with the same name as a primitive function of it (introduced long ago). Then the (old) primitive function suddenly seemed to give strange results so after this experience I will be careful about possible name collisions between record components and primitive functions.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Mon, 30 Dec 2019 17:16:17 -0600

> One could ask, why is such an ambiguity not rejected (made illegal)?

> Probably because such an illegality rule would have made many illegal many

> Ada programs that were legal before the introduction of the

> "object.operation" syntax for tagged-record objects.

The other reason is that there isn't any alternative notation available for components, whereas there is an alternative method for function calls. Ergo, we assume that you mean a component if both are available -- otherwise, it would be impossible to access a component at all if there is a function with the same name visible. Since that function wouldn't even have to be in the same scope, there would be a significant maintenance hazard.

Moral: This is another reason to make everything a private type (and also to not use prefixed notation with types that aren't private).

Generating Files with GPRbuild

From: mockturtle <framefritti@gmail.com>
Subject: gpr and Makefiles
Date: Mon, 27 Jan 2020 08:22:40 -0800
Newsgroups: comp.lang.ada

I have a question about the interaction between gprbuild and Makefile. I googled a bit and found mostly how to use gprbuild inside a Makefile, but, in a sense, I am interested in the other way around.

More precisely, among all my source files there is one package (say, foo.ads) that is actually generated by an external file (say, bar.txt) using a utility (call it "convert"). The matter is a bit more complex, but this is the core of the issue.

I can express the dependency between foo.ads and bar.txt in a Makefile like foo.ads: bar.txt

convert *--from=bar.txt --to=foo.ads*

What I would like is having gprbuild checking if bar.txt is newer than foo.ads; if it is, run convert and after that proceed with the actual building.

Is this possible?

I also checked Gem #152 (https://www.adacore.com/gems/gem-152-defining-a-new-language-in-a-project-file) about defining a new language inside a gpr file, but I am not sure it can be a solution.

Thank you in advance for your help

From: Shark8 <onewingedshark@gmail.com>
Date: Mon, 27 Jan 2020 09:49:36 -0800

> [...]

> I also checked Gem #152 [...] about defining a new language inside a gpr file, but I am not sure it can be a solution.

Why not?

Wouldn't you just use

```
package Compiler is
   for Driver ("Converter") use "convert";
   for Object_Generated ("Converter")
   use "False";
   --...
end Compiler;
```

From: mockturtle <framefritti@gmail.com>
Date: Mon, 27 Jan 2020 12:28:57 -0800

It worked, thank you.

Actually, it was less trivial than I expected. The main problem was that gprbuild expects a command line like

<compiler name> <pre-options> <source> <post-options>

while my command line was

convert <output filename> <input filename>

However, since convert is actually a Ruby script I changed it to handle the case <output>=-c as an "automagical" case where the output filename is obtained from the input.

From: briot.emmanuel@gmail.com
Date: Tue, 28 Jan 2020 03:57:51 -0800

gprbuild is pretty weak for generated code. When I was working at AdaCore, we had made a nice design to properly handle this, but I don't know what happened to that design.

Here, you are trying to generate Ada code. So when you start gprbuild, it might quickly compile a unit that depends on one of the generated Ada packages, without having generated them already. In practice, you end up having to run gprbuild multiple times (once to generate the files, then to compile everything). A proper build tool should be able to handle that automatically in one pass, just by having a full graph of dependencies.

Catching All Elaboration-Time Exceptions

From: ahlan@marriott.org
Subject: Last chance handler on a PC
Date: Thu, 30 Jan 2020 00:55:41 -0800
Newsgroups: comp.lang.ada

Does anyone know if it is possible to install a last chance handler for a PC program. i.e., write a procedure that gets called when a program issues an unhandled exception If it is possible how do you do it?

From: Egil H H <ehh.public@gmail.com>
Date: Thu, 30 Jan 2020 01:17:15 -0800

You can use the Termination_Handler in Annex C.7.3 (Added in Ada 2005), provided your compiler/runtime supports it.

http://www.ada-auth.org/standards/rm12_w_tc1/html/RM-C-7-3.html

Example usage is discussed in the Ada 2005 Rationale:

https://www.adaic.org/resources/add_content/standards/05rat/html/Rat-5-2.html#I1150

From: ahlan@marriott.org
Date: Thu, 30 Jan 2020 11:27:50 -0800

Very interesting but we want to catch all unhandled exceptions, specifically those raised during package elaboration.

From: ahlan@marriott.org
Date: Thu, 30 Jan 2020 11:35:56 -0800

To answer my own question...

To catch unhandled exceptions you only need to write a simple procedure and export it as __gnat_last_chance_handler.

This is linked into the program in preference to the default last chance handler provided by GNAT.

This procedure is called if nothing catches a raised exception.

Including those raised during package elaboration.

The procedure is not allowed to return so after doing whatever it is you want to do with the exception you must call __gant_unhandled_terminate

The following is an example.

```
procedure Last_Chance_Handler
(Occurrence :
Ada.Exceptions.Exception_Occurrence)
 with
   No_Return, Unreferenced, Export,
   Convention    => C,
   External_Name =>
    "__gnat_last_chance_handler";

 procedure Last_Chance_Handler
(Occurrence :
Ada.Exceptions.Exception_Occurrence) is
   procedure Unhandled_Terminate
   with
    No_Return, Import,
    Convention    => C,
    External_Name =>
    "__gnat_unhandled_terminate";

 begin
  begin
   null; -- Process the exception here.
  exception
  when others =>
   null;
  end;
  Unhandled_Terminate;
 end Last_Chance_Handler;
```

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Thu, 30 Jan 2020 21:02:17 +0100

Doing

```
Ada.Task_Termination.Set_Specific_Handler
  (T  =>
Ada.Task_Identification.Environment_Task,
   Handler => Last_Chance'access);
```

should do the same thing more portably. It will be called when the environment task terminates for any reason; you would only want it to actually do something when Cause = Unhandled_Exception.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Thu, 30 Jan 2020 22:26:38 +0200

Looks good, but to catch all elaboration-time exceptions (in other packages) the package that executes that call, in its own elaboration code, must be elaborated before all other packages. Do you have some easy way to ensure that, without inserting elaboration pragmas in all other packages?

I had a similar elaboration problem some time ago in an embedded application, where I wanted to set up some HW error-trap handlers that I would like to be active also during elaboration, but I found no easy way to ensure that the trap-handling package would be elaborated before all other packages.

From: "Jeffrey R. Carter" <spam.jrcarter.not@spam.not.acm.org>
Date: Thu, 30 Jan 2020 21:51:09 +0100

Of course that call has to be done before anything that might raise an exception during elaboration. Usually you'd put it in its own pkg, and then every other library-level unit in the system would with it with a pragma Elaborate_Body for it. If everything is part of a hierarchy, then only the spec of the root package of the hierarchy should need to do that.

Time Image and ARM Interpretations

From: Marius Amado-Alves <amado.alves@gmail.com>
Subject: Ada.Calendar.Formatting.Image (or Time_Of) changing the time
Date: Mon, 2 Mar 2020 10:49:52 -0800
Newsgroups: comp.lang.ada

Feeding Ada.Calendar.Formatting.Image with an Ada.Calendar.Time_Of the year, month, day, seconds on the left, we get the image on the right. Some images, marked *, are 1 hour behind.

2015 1 21 32040 (8:54 AM) => 2015-01-21 08:54:00

2015 1 21 39240 (10:54 AM) => 2015-01-21 10:54:00

2015 7 21 32040 (8:54 AM) => 2015-07-21 07:54:00 *

2015 7 21 39240 (10:54 AM) => 2015-07-21 09:54:00 *

The different input is the month, January versus July, so it looks like a daylight savings thing. Is this expected behaviour? Thanks.

[Compiler = GNAT Community 2018 (20180523-73)]

From: Simon Wright <simon@pushface.org>
Date: Tue, 03 Mar 2020 14:53:43 +0000

There was a conversation on Ada-Comment in June last year, in which it turned out that compiler implementers may have been misinterpreting the ARM. It was quite confusing.

Part of the problem is that Ada.Calendar.Clock, implemented over the OS facilities, may or may not be in local time; and how does it treat times which are not in the 'now' time zone?

[...]

From: Simon Wright <simon@pushface.org>
Date: Tue, 03 Mar 2020 17:40:06 +0000

Also, see AI95-00160,

http://www.ada-auth.org/cgi-bin/cvsweb.cgi/ais/ai-00160.txt?rev=1.4&raw=N

[This AI deals with the problem of times that happen twice at the boundaries of daylight savings time. —arm]

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Tue, 3 Mar 2020 17:49:31 -0600

> There was a conversation on Ada-Comment in June last year, in which it turned out that compiler implementers may have been misinterpreting the ARM. It was quite confusing.

Not just a conversation, but also a Binding Interpretation AI (which therefore applies to Ada 2012 compilers), AI12-0336-1.

Essentially, the formal definition of Time_Offset, and the way it was actually implemented by every compiler except mine, was completely different. We decided to match practice, especially as that matches the way the Internet uses time offsets. So that part of the RM was rewritten.

As Dmitry says, the default Time_Offset on GNAT gives one UTC. If you want CST or CDT (my time zones, which change on this coming Sunday), one needs to use -360 or -300. We've added a new renaming Local_Time_Image to make this relatively easy (dunno if GNAT has it yet).

The advantage of this definition is that the base UTC time doesn't jump during the year, but if you are interested in local time, you have to determine the offset based on the time-of-year.

Your example suggests that GNAT is doing something weird with times that are far away from today. That's certainly not intended, sounds like a bug to me. Certainly is a bug with the rewritten rules for Time_Image.

Enforcing Instantiations at Library Level

From: Vincent Marciante <vincent.marciante@l3harris.com>
Subject: Good/best way to enforce library-level instantiation a generic package
Date: Mon, 16 Mar 2020 11:51:25 -0700
Newsgroups: comp.lang.ada

I made a generic package that I want only to be instantiated at library level. I'm working on compile-time a way to enforce that desire which involves access type accessibility level checking but have not yet set it up correctly. Is there a better/standard way?

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Mon, 16 Mar 2020 20:21:16 -0500

For Ada 95, deriving from Controlled does the trick, but that was eliminated (at substantial cost) in Ada 2005 and later.

I suppose you could use type String_Access (which is a library-level access type) for this:

```
with Ada.Strings.Unbounded;
generic
   ...
package My_Generic is
   -- Real stuff here.
   Library-Level: constant aliased String
                     := "Library-Level";
   Check : Ada.Strings.
              Unbounded.String_Access :=
               Library_Level'Access;
   -- 'Access is illegal if My_Generic is not
   -- instantiated at the library level.
end My_Generic;
```

String_Access is a silly type that isn't used in the spec of Ada.Strings.Unbounded, and thus shouldn't be there, but it does work for this use.

You can of course use any library-level access type in your program for this purpose; I picked this one 'cause it is already sitting around.

From: briot.emmanuel@gmail.com
Date: Mon, 16 Mar 2020 23:29:56 -0700

The way I do this is using gnat-specific pragmas and attributes:

```
generic
package Generics is
   pragma Compile_Time_Error
     (not Generics'Library_Level,
          "must be at library level");
   ...
end Generics;
```

From: Vincent Marciante <vincent.marciante@l3harris.com>
Date: Tue, 17 Mar 2020 03:15:14 -0700

'Library_Level is nice and clean! It should be part of the Standard! I am using GNAT but still have to be compatible with other compiler so will have to go with something along the lines of Randy's suggestion.