ADA USER JOURNAL

Volume 41
Number 1
March 2020

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

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

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

Proceedings and Special Issues

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

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

News and Product Announcements

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

Reprinted Articles

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

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

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

Editorial

This is a very special and peculiar issue of the AUJ, for several reasons.

The first issue of the Ada User Journal (in fact, of the Ada UK News, which preceded the AUJ) was published in March 1980, which means that in March 2020 we celebrated its 40th anniversary (for the most curious about historical facts, I recommend taking a look at the "History" page of the AUJ on https://www.ada-europe.org/auj/history/). Therefore, this is a special "Anniversary Issue", featuring specific contents to adequately celebrate the occasion.

While we were commemorating the AUJ anniversary, the world was struck by the COVID-19 pandemic. Many of us could keep working remotely, but many events and activities were cancelled or postponed, and many more were affected in various ways. In particular, the 25th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2020) had to be cancelled and postponed to June 2021, although with different repercussions on the conference elements, as detailed in the Notice of Cancellation on page 29. The preparation of this issue was not an exception – we missed deadlines, postponed tasks, and eventually were unable to complete its preparation in due time. The large delay, certainly a rare event, makes it a peculiar issue, even if not for good reasons.

Finally, and still a direct consequence of this extraordinary situation that we are now living, we were (and still are) unable to print and post the AUJ to our subscribers. We thus decided to make it available in a digital form, to avoid any further delays, with a promise that the printed copy will follow later, whenever businesses reopen, and we can resume the normal procedures. Being fully and publicly available on the AUJ online archive without any embargo period, also makes this AUJ issue very peculiar.

In this special Anniversary issue, we include three invited articles with which we intend to somehow celebrate the relevance of the Ada language and important events that took place in the last decade. We start with an article by John Barnes, entitled "From Byron to the Ada Language", which is a polished version of an article that appeared in the AUJ December 2015 issue, to celebrate the 200th birthday of Ada Lovelace. We then include an article by Carl Brandon that describes the fundamental role of the Ada language in building an extremely reliable system – a CubeSat that worked. The third invited article is authored by Benjamin Brosgol, reporting the AdaCore experience and lessons learned in the development of open source software that is commercially viable.

Then, the issue includes the fourth article on the series of reports on the work of the Ada Rapporteur Group (ARG), written by Jeff Cousins, member and former chair of the ARG, which provides further updates on the proposed changes for the next edition of Ada.

After that, we conclude the publication of the Proceedings of the Workshop on Challenges and New Approaches for Dependable and Cyber-Physical Systems Engineering (DeCPS 2019), with a paper co-authored by researchers from CISTER/ISEP (Portugal), Ikerlan Technology Research Centre (Spain) and Thales (France). The paper is about the ELASTIC software architecture for processing data from many sources while satisfying non-functional requirements related to real-time, security or energy-efficiency, highlighting the constraints imposed to the architecture that are necessary to fulfil those requirements.

Finally, we close this anniversary issue with some thoughts by John Barnes on memorable Ada-Europe conferences over the past 40 years, along with a puzzle that would have probably been given at this years' conference. We hope the readers will enjoy as they would have enjoyed during the conference.

Last but not the least, the issue includes, as usual, the Quarterly News Digest and Calendar sections, prepared respectively by Alejandro R. Mosteo and Dirk Craeynest, their editors.

Antonio Casimiro
Lisboa
March 2020
Email: AUJ_Editor@Ada-Europe.org

Quarterly News Digest

Alejandro R. Mosteo

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

Contents

[Messages without subject/newsgroups are replies from the same thread. Messages may have been edited for minor proofreading fixes.
—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.

Conference Calendar

Dirk Craeynest

KU Leuven, Belgium. Email: Dirk.Craeynest@cs.kuleuven.be

This is a list of European and large, worldwide events that may be of interest to the Ada community. Further information on items marked ♦ is available in the Forthcoming Events section of the Journal. Items in larger font denote events with specific Ada focus. Items marked with ☺ denote events with close relation to Ada.

The information in this section is extracted from the on-line *Conferences and events for the international Ada community* at http://www.cs.kuleuven.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programs, URLs, etc. and are updated regularly.

The COVID-19 pandemic had a catastrophic impact on conferences world-wide. Where available, the status of events is indicated with the following markers: "(c)" = event cancelled, "(p)" = event postponed to a later date, and "(v)" = event is fully or partially held online.

2020

April 01-03 (p) — 20th **International Real-Time Ada Workshop** (IRTAW'2020). Benicàssim, Spain. In cooperation with Ada-Europe.

April 15-17 (p) — 24th **International Conference on Evaluation and Assessment in Software Engineering** (EASE'2020). Trondheim, Norway. EASE'2020 was postponed from 15-17 April 2020 to June 2021. Topics include: assessing the benefits / costs associated with using chosen development technologies; empirical studies using qualitative, quantitative, and mixed methods; evaluation and comparison of techniques and models; replication of empirical studies and families of studies; etc.

April 21-24 (v) — 13th **Cyber-Physical Systems and Internet of Things Week** (CPS Week'2020). Sydney, Australia. Event includes: 4 top conferences, HSCC, ICCPS, IPSN, and RTAS, as well as up to 20 satellite events, such as poster and demo sessions, workshops, tutorials, competitions, industrial exhibitions, and summit forums.

- ☺ April 21-24 — 26th IEEE **Real-Time and Embedded Technology and Applications Symposium** (RTAS'2020). Topics include: research related to embedded systems or timing issues, ranging from traditional hard real-time systems to embedded systems without explicit timing requirements, including latency-sensitive systems with informal or soft real-time requirements; original systems and applications, case studies, methodologies, and applied algorithms that contribute to the state of practice in the design, implementation, verification, and validation of embedded systems and time-sensitive systems (of any size); etc.

April 25-30 (p) — 23rd **European Joint Conferences on Theory and Practice of Software** (ETAPS'2020). Dublin, Ireland. ETAPS'2020 was postponed from 25-30 April 2020 to fall 2020.

- ☺ April 25-30 — **VerifyThis Verification Competition 2020**. Topics include: VerifyThis Collaborative Long-Term Challenge; no restrictions on verification technology used; at least one team is using SPARK.
- ☺ April 26 — 12th **Workshop on Programming Language Approaches to Concurrency- and communication-cEntric Software** (PLACES'2020). Topics include: general area of programming language approaches to concurrency, communication, and distribution and may range from foundational issues, language implementations, to applications and case studies; design and implementation of programming languages with first class concurrency and communication; models, such as process algebra and automata; concurrent data types, objects, and actors; verification and program analysis methods for concurrent and distributed software; etc.

April 27-30 (v) — 15th **European Conference on Computer Systems** (EuroSys'2020). Heraklion, Crete, Greece. Topics include: all areas of computer systems research, such as distributed systems, language support and runtime systems, systems security and privacy, dependable systems, parallelism, concurrency, and multicore systems, real-time, embedded, and cyber-physical systems, tracing, analysis, verification, and transformation of systems, etc.

May 11-13 (c) ACM **International Conference on Computing Frontiers** 2020 (CF'2020). Catania, Sicily, Italy. Topics include: embedded, IoT and cyber-physical systems; large-scale system design and networking; system software, compiler technologies and programming languages; fault tolerance and resilience; security; etc.

May 11-15 (v) 12th **NASA Formal Methods Symposium** (NFM'2020). Moffett Field, California, USA. Topics include: identifying challenges and providing solutions towards achieving assurance for critical systems; formal verification, including theorem proving, model checking, and static analysis; run-time verification; techniques and algorithms for scaling formal methods, such as abstraction and symbolic methods, compositional techniques, as well as parallel and/or distributed techniques; safety cases and system safety; formal approaches to fault tolerance; design for verification and correct-by-design techniques; empirical evaluations of formal methods techniques for safety-critical systems; formal methods in systems engineering and model-based development; etc.

☺ May 19-21 (v) 23rd IEEE **International Symposium On Real-Time Distributed Computing** (ISORC'2020). Nashville, Tennessee, USA. Topics include: object/component/service-oriented real-time distributed computing (ORC) technology; software architectures for real-time distributing computing (programming paradigms, ORC paradigms, object/component models, languages, synchronous languages), trusted and dependable systems, system software (real-time kernels, operating systems, distribution middleware for ORC, extensibility, synchronization, resource allocation, scheduling, timing analysis, fault tolerance and resilience, security, ...), applications (medical devices, intelligent transportation systems, industrial automation systems and Industry 4.0, Internet of Things and Smart Grids, embedded and cyber-physical systems in automotive, avionics, autonomous vehicles, consumer electronics, ...), system evaluation (performance & timing evaluation, dependability, fault detection and recovery time, ...), etc.

May 25-26 23rd **International Workshop on Software and Compilers for Embedded Systems** (SCOPES'2020) St. Goar, Germany. Topics include: all aspects of compilation and mapping process of embedded systems, such as models of computation and programming languages; automatic code parallelization techniques; mapping and scheduling techniques for embedded multi-processor systems; code generation techniques for embedded single- and multi-processor architectures; design of real-time systems; techniques for compiler aided profiling, measurement, debugging and validation of embedded software; etc.

May 26 **Ada-France Webinar**. Topics include (in French): découvrir le langage Ada et ses atouts pour vos développements logiciels.

June 03-05 (c) 20th **International Conference on Computational Science** (ICCS'2020). Amsterdam, the Netherlands. Topics include: scientific computing, complex systems - modelling and simulation, parallel and distributed computing, new programming models, education in computational science, etc.

♦ June 08-12 (p) 25th **Ada-Europe International Conference on Reliable Software Technologies** (AEiC 2020 aka Ada-Europe 2020). Santander, Spain. AEiC'2020 was postponed from 8-12 June 2020 to June 2021. Sponsored by Ada-Europe, in cooperation with ACM SIGAda, SIGBED, and the Ada Resource Association (ARA).

June 08-12 (v) 21st **International Conference on Agile Software and Systems Development** (XP'2020). Copenhagen, Denmark.

☺ June 09-11 (v) 28th **International Conference on Real-Time Networks and Systems** (RTNS'2020). Paris, France. Topics include: real-time applications design and evaluation (automotive, avionics, space, railways, telecommunications, process control, multimedia), real-time aspects of emerging smart systems (cyber-physical systems and emerging applications, ...), real-time system design and analysis (real-time tasks modeling, task/message scheduling, mixed-criticality systems, Worst-Case Execution Time (WCET) analysis, ...), software technologies for real-time systems (model-driven engineering, programming languages, compilers, WCET-aware compilation and parallelization strategies, middleware, Real-time Operating Systems (RTOS), hypervisors, ...), formal specification and verification, real-time distributed systems (fault tolerance, publisher/subscriber protocols, ...), etc.

☺ June 16 (v) — 21st ACM SIGPLAN/SIGBED **International Conference on Languages, Compilers, and Tools for Embedded Systems** (LCTES'2020). London, UK. Topics include: support for enhanced programmer productivity; support for enhanced debugging, profiling, and exception/interrupt handling; hardware, system software, application software, and their interfaces; system integration and testing; run-time system support for embedded systems; support for system security and system-level reliability; validation and verification, in particular of concurrent and distributed systems; formal foundations of model-based design for code generation, analysis, verification; architecture for new language features, virtualization, compilation, debugging tools; empirical studies and their reproduction, and confirmation; etc.

June 18 (v) — **European Conference on Software Engineering Education** (ECSEE'2020). Seeon Monastery, Bavaria, Germany.

June 22-26 (p) — **Software Technologies: Applications and Foundations** (STAF'2020). Bergen, Norway. STAF'2020 was postponed from 22-26 June 2020 to June 2021.

> June 22-26 — 14th **International Conference on Tests And Proofs** (TAP'2020). Topics include: many aspects of verification technology, including foundational work, tool development, and empirical research; the connection between proofs (and other static techniques) and testing (and other dynamic techniques); verification and analysis techniques combining proofs and tests; program proving with the aid of testing techniques; deductive techniques supporting the automated generation of test vectors and oracles; deductive techniques supporting novel definitions of coverage criteria; program analysis techniques combining static and dynamic analysis; testing and runtime analysis of formal specifications; verification of verification tools and environments; applications of test and proof techniques in new domains, such as security, configuration management, learning; combined approaches of test and proof in the context of formal certifications (Common Criteria, CENELEC, ...); case studies, tool and framework descriptions, and experience reports about combining tests and proofs; etc.

June 23-26 (v) — 26th **International Working Conference on Requirements Engineering: Foundation for Software Quality** (REFSQ'2020). Pisa, Italy. REFSQ'2020 was postponed from 24-27 March to a virtual event on 23-26 June.

July 06 - 11 (v) — 42nd **International Conference on Software Engineering** (ICSE'2020). Seoul, South Korea. ICSE'2020 was postponed from 23-27 May to June 27 – July 19. Topics include: the full spectrum of Software Engineering.

> July 7-10 — **Software Engineering Education and Training** (SEET'2020). Topics include: novel methods of teaching software engineering skills, empirical studies describing software engineering education contexts, novel learning technologies that support software engineering education and training, well-substantiated arguments about what skills are most essential to learn, etc.

> July 13 — 8th **International Conference on Formal Methods in Software Engineering** (FormaliSE'2020). Topics include: approaches and tools for verification and validation; application of formal methods to specific domains, e.g. autonomous, cyber-physical, and IoT systems; scalability of formal methods applications; integration of formal methods within the software development lifecycle formal specification; model-based engineering approaches; formal methods in a certification context; formal approaches for safety and security-related issues; usability of formal methods; guidelines to use formal methods in practice; case studies developed/analyzed with formal approaches; experience reports on the application of formal methods to real-world problems; etc.

> June 28-30 — 3rd **International Conference on Technical Debt** (TechDebt'2020). Topics include: the business case for technical debt management; understanding causes and effects of technical debt; technical debt management within software life-cycle management; technical debt in design and architecture; technical debt and software evolution, maintenance, and aging; concrete practices and tools used to manage technical debt; debt remediation and refactoring; technical debt and quality attributes, such as security (especially at run-time); technical debt in (ultra-) large-scale systems, ecosystems, platforms and product lines; success and failure stories of technical debt management; education and training related to technical debt; etc.

July 07-10 (v)	32nd **Euromicro Conference on Real-Time Systems** (ECRTS'2020). Modena, Italy.
	July 07 5th **Workshop on Security and Dependability of Critical Embedded Real-Time Systems** (CERTS'2020). Topics include: the intersection of security and dependability of embedded and real-time systems, with an emphasis on criticality and distribution. Deadline for submissions: April 16, 2020.
☺ July 13-17	34th **European Conference on Object-Oriented Programming** (ECOOP'2020). Berlin, Germany. Topics include: design, implementation, optimization, analysis, and theory of programs, programming languages, and programming environments.
July 13-17 (v)	44th **Annual** IEEE **Conference on Computers, Software and Applications** (COMPSAC'2020). Madrid, Spain. Deadline for submissions: April 9, 2020 (workshop papers).
July 13-17 (v)	14th ACM/IFIP **International Conference on Distributed Event-Based Systems** (DEBS'2020). Montreal, Quebec, Canada. Topics include: systems dealing with collecting, detecting, processing and responding to events through distributed middleware and applications; embedded systems, real-time analytics, complex event processing, distributed programming, security, reliability and resilience, Internet-of-Things, cyber-physical systems, etc. Deadline for submissions: May 10, 2020 (posters, demos, doctoral symposium papers).
☺ August 19-21	26th IEEE **International Conference on Embedded Real-Time Computing Systems and Applications** (RTCSA'2020). Gangnueng, South Korea. Topics include: real-time scheduling, timing analysis, programming languages and run-time systems, middleware systems, applications and case studies of IoT and CPS, cyber-physical co-design, multi-core embedded systems, fault tolerance and security, etc. Deadline for submissions: April 14, 2020 (papers), June 1, 2020 (student session).
August 26-28	46th **Euromicro Conference on Software Engineering and Advanced Applications** (SEAA'2020). Portoroz, Slovenia. Topics include: information technology for software-intensive systems; conference tracks on Embedded Systems and the Internet of Things (ES-IoT), Software Process and Product Improvement (SPPI), Model-Driven Engineering and Modeling Language (MDEML), etc.; special sessions on Cyber-Physical Systems (CPS), Software Engineering and Technical Debt (SEaTeD), etc.
September 01-04 (v)	31st **International Conference on Concurrency Theory** (CONCUR'2020). Vienna, Austria. Topics include: semantics, logics, verification and analysis of concurrent systems; basic models of concurrency; verification and analysis techniques for concurrent systems such as abstract interpretation, atomicity checking, model checking, race detection, run-time verification, static analysis, testing, theorem proving, type systems, security analysis, ...; distributed algorithms and data structures; theoretical foundations of architectures, execution environments, and software development for concurrent systems such as multiprocessor and multi-core architectures, compilers and tools for concurrent programming, programming models such as component-based, object-oriented, ...; etc. Deadline for submissions: April 28, 2020 (abstracts), May 6, 2020 (papers).
September 02-03 (v)	25th **International Conference on Formal Methods for Industrial Critical Systems** (FMICS'2020). Vienna, Austria. Co-located with CONCUR'2020 and FORMATS'2020. Topics include: case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; methods, techniques and tools to support automated analysis, certification, debugging, descriptions, learning, optimisation and transformation of complex, distributed, real-time, embedded, mobile and autonomous systems; verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability (e.g., scalability and usability issues); impact of the adoption of formal methods on the development process and associated costs; application of formal methods in standardisation and industrial forums. Deadline for submissions: May 8, 2020 (abstracts), May 15, 2020 (papers).
September 06-09	15th **Federated Conference on Computer Science and Information Systems** (FedCSIS'2020). Sofia, Bulgaria. Event includes: Language Technologies and Applications (5th Workshop LTA'20), Scalable Computing (11th Workshop WSC'20), Cyber Security, Privacy and Trust (1st International Forum NEMESIS'20), Advances in Software and System Engineering (ASSE'20), Cyber-Physical Systems (7th Workshop IWCPS'20), Lean and Agile Software Development (4th International Conference LASD'20), Model Driven Approaches in System Development (6th Workshop MDASD'20), Software Engineering (40th IEEE Workshop SEW'20), etc. Deadline for submissions: May 15, 2020 (papers), June 9, 2020 (position papers).

September 09-11 13th **International Conference on the Quality of Information and Communications Technology** (QUATIC'2020). Faro, Portugal. Topics include: all quality aspects in ICT systems engineering and management; quality in ICT process, product and applications domains; practical studies; etc. Tracks on ICT verification and validation, safety, security and privacy, model-driven methods, agile methods, evolution in ICT / reengineering and refactoring, evidence-based software quality engineering, software quality education and training, etc. Deadline for submissions: May 25, 2020 (short papers).

September 14-18 (v) 18th **International Conference on Software Engineering and Formal Methods** (SEFM'2020). Amsterdam, the Netherlands. Topics include: software development methods (software evolution, maintenance, re-engineering, and reuse ...), design principles (programming languages, abstraction and refinement, ...), software testing, validation, and verification (model checking, theorem proving, and decision procedures; testing and runtime verification; other light-weight and scalable formal methods; ...), security and safety (security, privacy, and trust; safety-critical, fault-tolerant, and secure systems; software certification), applications and technology transfer (real-time, hybrid, and cyber-physical systems; education; ...), case studies, best practices, and experience reports. Deadline for submissions: April 27, 2020 (abstracts), May 4, 2020 (papers).

September 15-18 39th **International Conference on Computer Safety, Reliability and Security** (Safecomp'2020). Lisbon, Portugal. Topics include: all aspects related to the development, assessment, operation and maintenance of safety-related and safety-critical computer systems; formal modelling, verification and validation; model-driven engineering; security and privacy protection mechanisms; safety/security co-engineering and risk assessment; testing, verification and validation methods & tools; qualification, assurance and certification methods & tools; cyber-physical threats and vulnerability analysis; safety and security guidelines, standards and certification; etc. Domains of application include: railways, automotive, space, avionics & process industries; highly automated and autonomous systems; telecommunication and networks; safety-related applications of smart systems and IoT; critical infrastructures; medical devices and healthcare; surveillance, defense, emergency & rescue; logistics, industrial automation, off-shore technology; education & training; etc. Deadline for submissions: May 28, 2020 (position papers).

September 20-25 **Embedded Systems Week** 2020 (ESWEEK'2020). Hamburg, Germany. ESWEEK'2020 was moved from October 11-16 in Shanghai, China, to September 20-25 in Hamburg, Germany. Deadline for submissions: April 3, 2020 (Journal Track abstracts), April 17, 2020 (Journal Track full papers, workshops), May 1, 2020 (tutorials, special sessions), June 5, 2020 (Work-in-Progress papers), July 13, 2020 (PhD Student Forum extended abstracts).

> Sep 20-25 **International Conference on Compilers, Architecture, and Synthesis for Embedded Systems** (CASES'2020). Topics include: latest advances in compilers and architectures for high-performance, low-power, and domain-specific embedded systems; compilers for embedded systems; multi- and many-core processors, GPU architectures, reconfigurable computing including FPGAs and CGRAs; security, reliability, and predictability (secure architectures, hardware security, and compilation for software security; architecture and compiler techniques for reliability and aging; modeling, design, analysis, and optimization for timing and predictability; validation, verification, testing & debugging of embedded software); Trusted IoT Day; etc.
>
> Sep 20-25 **International Conference on Hardware/Software Codesign and System Synthesis** (CODES+ISSS'2020). Topics include: system-level design, hardware/software co-design, modeling, analysis, and implementation of modern embedded and cyber-physical systems, from system-level specification and optimization to system synthesis of multi-processor hardware/software implementations.
>
> Sep 20-25 ACM SIGBED **International Conference on Embedded Software** (EMSOFT'2020). Topics include: the science, engineering, and technology of embedded software development; research in the design and analysis of software that interacts with physical processes; results on cyber-physical systems, which compose computation, networking, and physical dynamics.

September 21-24 39th **International Symposium on Reliable Distributed Systems** (SRDS'2020). Shanghai, China. Topics include: distributed systems design, development and evaluation, with emphasis on reliability, availability, safety, dependability, security, and real-time. Deadline for submissions: April 27, 2020 (workshops), May 8, 2020 (abstracts), May 15, 2020 (full papers).

September 21-25	35th IEEE/ACM **International Conference on Automated Software Engineering** (ASE'2020). Melbourne, Australia. Topics include: foundations, techniques, and tools for automating the analysis, design, implementation, testing, and maintenance of large software systems; testing, verification, and validation; software analysis; empirical software engineering; maintenance and evolution; software security and trust; program comprehension; software architecture and design; reverse engineering and re-engineering; model-driven development; specification languages; software product line engineering; etc. Deadline for submissions: April 17, 2020 (research track abstracts), April 24, 2020 (research track papers, tutorials), May 29, 2020 (most other tracks).
September 22-24	19th **International Conference on Intelligent Software Methodologies, Tools and Techniques** (SOMET'2020). Kytakyushu, Japan. Topics include: state-of-art and new trends on software methodologies, tools and techniques; software methodologies, and tools for robust, reliable, non-fragile software design; software developments techniques and legacy systems; automatic software generation versus reuse, and legacy systems; software evolution techniques; Agile Software and Lean Methods; formal methods for software design; software maintenance; software security tools and techniques; formal techniques for software representation, software testing and validation; software reliability; Model Driven Development (DVD), code centric to model centric software engineering; etc.
October 06-09	20th **International Conference on Runtime Verification** (RV'2020). Los Angeles, California, USA. Topics include: monitoring and analysis of the runtime behaviour of software and hardware systems. Application areas include cyber-physical systems, safety/mission critical systems, enterprise and systems software, cloud systems, autonomous and reactive control systems, health management and diagnosis systems, and system security and privacy. Deadline for submissions: May 18, 2020 (abstracts), May 25, 2020 (papers, tutorials).
October 24-28	13th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2020). Porto, Portugal. ICST'2020 was postponed from 23-27 March to 24-28 October. Topics include: manual testing practices and techniques, security testing, model based testing, test automation, static analysis and symbolic execution, formal verification and model checking, software reliability, testability and design, testing and development processes, testing in specific domains (such as embedded, concurrent, distributed, ..., and real-time systems), testing/debugging tools, empirical studies, experience reports, etc.
November 08-13	28th ACM **Joint European Software Engineering Conference** and **Symposium on the Foundations of Software Engineering** (ESEC/FSE'2020). San Francisco, California, USA. Topics include: agile software development; component-based software engineering; configuration management and deployment; cyber physical systems; debugging; dependability, safety, and reliability; education; embedded software; emerging domains of software; empirical software engineering; formal methods; middleware, frameworks, and APIs; mining software engineering repositories; model-driven engineering; parallel, distributed, and concurrent systems; program analysis; program comprehension; program repair; programming languages; refactoring; reverse engineering; safety-critical systems; scientific computing; security, privacy and trust; software architecture; software economics and metrics; software evolution and maintenance; software modeling and design; software process; software product lines; software reuse; software testing; software visualization; specification and modeling languages; tools and environments; traceability; validation and verification; etc. Deadline for submissions: April 1, 2020 (workshops), May 5, 2020 (industry - full papers), June 19, 2020 (doctoral symposium, tool demos), June 26, 2020 (student research competition), June 30, 2020 (Journal First - full papers).
November 09-11	19th **International Conference on Software Reuse** (ICSR'2020). Hammamet, Tunisia. Theme: "Reuse in emerging software engineering practices". Topics include: new and innovative research results and industrial experience reports dealing with all aspects of software reuse within the context of the modern software development landscape. Deadline for submissions: June 19, 2020 (research paper abstracts), July 3, 2020 (full papers).
November 09-12	32nd **International Conference on Software Engineering Education and Training** (CSEET'2020). Munich, Germany. CSEET'2020 was postponed from 28-31 July to 9-12 November. Topics include: Teaching formal methods (TFM), Teaching "real world" SE practices (TRW), Software quality assurance education (SQE), Global and distributed SE education (GDE), Open source in education (OSE), Cooperation between Industry and Academia (CIA), Training models in industry (TMI), Continuous education (CED), Methodological aspects of SE education (MAE), etc.
☺ November 15-20	ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2020). Chicago, USA. Topics include: all aspects of software construction, at the

	intersection of programming, languages, and software engineering. Deadline for submissions: April 1, 2020 (SPLASH workshops), April 15, 2020 (PACMPL Issue OOPSLA), April 22, 2020 (Static Analysis Symposium abstracts), April 23, 2020 (Onward! papers), April 24, 2020 (Static Analysis Symposium papers), May 23, 2020 (Onward! essays), June 11, 2020 (DLS - Dynamic Languages Symposium), June 21, 2020 (GPCE abstracts - Generative Programming: Concepts & Experiences, SLE abstracts - Software Language Engineering), June 28, 2020 (GPCE papers - Generative Programming: Concepts & Experiences, SLE papers - Software Language Engineering), July 10, 2020 (SPLASH-E), July 15, 2020 (Doctoral Symposium, Student Research Competition abstracts), Augustus 7, 2020 (PLMW Travel Grant Applications - PL Mentoring Workshop), Augustus 10, 2020 (posters), September 1, 2020 (student volunteer applications).
November 16-20	23rd **Ibero-American Conference on Software Engineering** (CIbSE'2020). Curitiba, Brazil. CIbSE'2020 was postponed from 4-8 May to 16-20 November.
November 25-27	21st **International Conference on Product-Focused Software Process Improvement** (PROFES'2020). Turin, Italy. Topics include: experiences, ideas, innovations, as well as concerns related to professional software development and process improvement driven by product and service quality needs. Deadline for submissions: June 5, 2020 (full research paper abstracts, workshops), June 12, 2020 (full research papers), July 31, 2020 (short papers, industry papers, tutorials), August 7, 2020 (Journal-First papers).
December 11-14	20th IEEE **International Conference on Software Quality, Reliability and Security** (QRS'2020). Macau, China. QRS'2020 was postponed from 27-31 July to December 11-14. Topics include: reliability, security, availability, and safety of software systems; software testing, verification, and validation; program debugging and comprehension; fault tolerance for software reliability improvement; modeling, prediction, simulation, and evaluation; metrics, measurements, and analysis; software vulnerabilities; formal methods; operating system security and reliability; benchmark, tools, industrial applications, and empirical studies; etc. Deadline for submissions: April 10, 2020 (regular and short papers), May 15, 2020 (workshop papers), June 10, 2020 (fast abstracts, industry track, posters).
☺ December 01-04	41st IEEE **Real-Time Systems Symposium** (RTSS'2020). Houston, Texas, USA. Deadline for submissions: July 2, 2020 (submissions).
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

2021

January 18-20	16th **International Conference on High Performance and Embedded Architecture and Compilation** (HiPEAC'2021). Budapest, Hungary. Topics include: computer architecture, programming models, compilers and operating systems for embedded and general-purpose systems. Deadline for submissions: June 30, 2020 (workshops, tutorials).
June 07-11	25th **Ada-Europe International Conference on Reliable Software Technologies** (AEiC 2021 aka Ada-Europe 2021). Santander, Spain. AEiC'2020 was postponed from 8-12 March 2020 to June 2021. Sponsored by Ada-Europe.
October 10-15	**Embedded Systems Week** 2021 (ESWEEK'2021). Shanghai, China. The venues for ESWEEK 2020 and 2021 were swapped. ESWEEK 2020 will now be held in Hamburg, Germany from September 20-25, 2020, and ESWEEK 2021 will be held in Shanghai, China from October 10-15, 2021.
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

Join Ada-Europe!

Become a member of Ada-Europe and **support Ada-related activities** and the future **development of the Ada programming language**.

Membership benefits include **receiving the quarterly Ada User Journal** and a substantial **discount when registering for the annual Ada-Europe conference**.

To apply for membership, visit our web page at

http://www.ada-europe.org/**join**

25th Ada-Europe
International Conference on
Reliable Software Technologies
(AEiC 2020)
8-12 June 2020, Santander, Spain
(C) RMR

Saturday, 21 March 2020

AEiC 2020 – Notice of Cancellation

The containment measures put in place on a world scale against the COVID-19 pandemic, which is affecting us all so severely, have taken precedence in everybody's personal and professional life. The consequent regulations issued by national governments and public health authorities in the regard of public gatherings, oblige us to **cancel the Ada-Europe International Conference 2020 (AEiC 2020)**, scheduled for *8-12 June 2020, in Santander, Spain*.

Consequent to that obligation, the organizing committee and its principal sponsor Ada-Europe have made a number of decisions, which we wish to share with authors, sponsors, participants and any other interested parties.

In regard to the calendar of events promoted by Ada-Europe:

- *Santander, Spain*, will be the venue of the next Ada-Europe International Conference on Reliable Software Technologies in **June 2021.**
- The venue originally earmarked for the 2021 edition of the AEiC, *Ghent, Belgium*, will be the prime candidate to host the **2022** edition of the conference.
- Selected elements of the technical program of the 2020 conference program will be considered for presentation at an *Ada event* co-organized with the International Real-Time Ada Workshop (IRTAW 2020), now tentatively postponed to **late October 2020, near Valencia, Spain**. The details of that event will be announced in due course, contingent on the lifting of current sanitary restrictions.

In regard to the elements of the 2020 conference program:

- The processing of the Journal-track submissions continues seamlessly, with a view to ensuring prompt open-access publication to those that gain acceptance. The authors of the accepted papers will be invited to present their work at the 2021 conference.
- The authors of the Industrial Presentations that are accepted after review will be invited to forward their work to the Ada User Journal for immediate publication.
- The call for Work-in-Progress submissions is cancelled. The prospective authors are invited to forward their work to the Ada User Journal for consideration.

We express our heartfelt friendship to all members of our community and their families in these difficult times, and look forward to the restoration of a safe normality.

See you all soon!

The Ada-Europe Board

Tools to get you there. Safely.
Ada and The GNAT Pro High-Integrity Family
www.adacore.com
AdaCore
The GNAT Pro Company

From Byron to the Ada Language

John Barnes
11 Albert Road, Caversham, Reading, RG4 7AN, UK; Tel: +44 118 947 4125; email: john@jbinformatics.co.uk

Abstract

This is a slightly polished version of an extended version of a talk given by the author at a Symposium in Oxford to celebrate the 200th birthday of Ada Lovelace in December 2015. It starts with a few words about Byron and his Bear at Cambridge. This is followed by some remarks about Babbage who also went to Cambridge and was involved in an important event involving Brunel and Safety on the Railways. The main topic is a look at the events that led to the new programming language devised to satisfy the needs of the US Department of Defense and which was named Ada in recognition of the fact that Ada Lovelace was the world's first programmer. The polishing is in celebration of 40 years of the Ada User Journal.

Keywords: Byron, Babbage, Ada.

1 Byron and his Bear

Byron went up to Trinity College, Cambridge in July 1805. He graduated as an MA in 1808. However, as a Noble he did not have to take any examinations in order to graduate but just had to stay in residence for the required number of terms (presumably nine as it is now). He did not graduate with honours as a BA as would happen to mere commoners but jumped straight to MA.

Incidentally, an important examination at Cambridge is known as a tripos. Thus the examination for mathematics is the Maths Tripos. The name derives from the fact that originally the examination started with a viva and the student sitting on a three-legged stool.

Another curiosity of the time was that one had to take the Maths Tripos before any other. Thus if one wanted to read Classics or Theology then one had to pass the Maths Tripos with honours first. Seems very sensible to me. Apart from Classics and Theology there probably wasn't anything else one could read. The sciences had not been discovered much so there was no Natural Philosophy. But the Maths Tripos did cover a lot of mechanics and optics (remember that Newton was keen on mechanics and optics).

Byron wanted to have a dog in college but dogs were (and still are) forbidden. Other animals are permitted and cats are common. So Byron decided that he wanted a Bear. It is quite clear that he did indeed have a Bear but there is some doubt about where Bruin was kept. Now according to John M F Wright who came up to Trinity in 1813 and writing much later in his anonymously published book, *Alma Mater*, in 1827, he says that Byron kept Bruin in Great Court. And then

"When Lord Byron was at Trinity, he kept in rooms on this staircase, round which you might drive a coach and six, and had, moreover, the use of the small Hexagonal one in the tower."

He is referring to K staircase in Great Court which has a tower/turret which contains a spiral staircase. K staircase and the diagonally opposite A staircase are in corners of the court. Figure 1 is a view of Great Court from the foot of K staircase showing clearly the opposite A staircase and its tower. Note the fountain in the centre of the court; the Master's lodge is to the left of A staircase. Figure 2 shows K staircase (with roadworks).

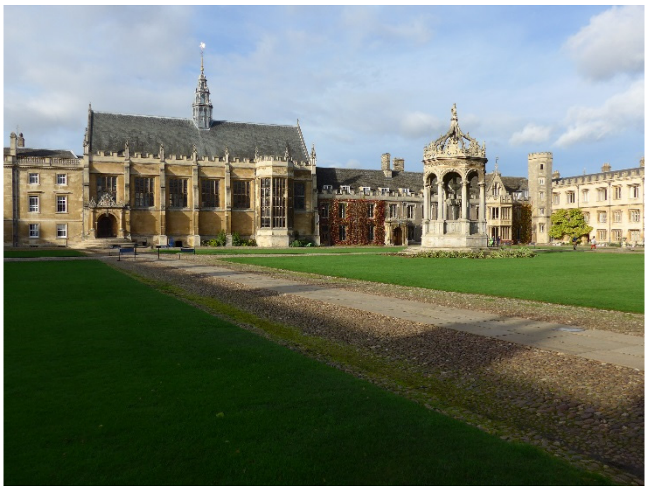


Figure 1. Great Court from K staircase

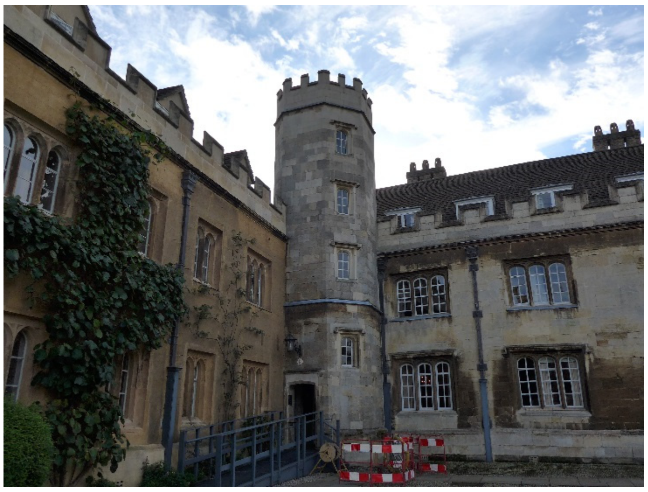


Figure 2. The turret at K staircase

The "hexagonal" room Wright refers to is at the top of the tower. If Byron did live on K staircase then he probably lived in a room on the first floor. Indeed K6 (with window open in the figure) is the best room although it would be tricky to drive a coach and six around it!

About 150 years later, by a strange coincidence, I was an undergraduate at Trinity, Cambridge and in my last year, I lived in Great Court in K6. I was told that I could well be living in Byron's old rooms. However, there was no sign of bears having been in the top of the turret which is now a toilet.

Further evidence is suggested by an item in a small book entitled *The Night Climber's Guide to Trinity* published in 1960. The K corner is known as Mutton-Hole corner for some reason and the book gives guidance on clambering on the roof around it. It says "The Mutton-hole Trail is a long stretch of leads running to the tower of the same name, where Byron once kept his bear." Moreover, this item is prefaced by

"Lo! dusky masses steal in dubious sight
Along the leaguered wall."

Byron, Don Juan, Canto VII

However, I am told by the college archivist that this story regarding the bear is almost certainly not true. As a Noble he would have had extra posh rooms and it is thought that he probably lived in I1 Nevile's Court. Posh but cold. But the bear was real and probably lived in Rams yard where Byron kept his horses.

It is a mystery to me as to why Wright wrote a fanciful story about the bear. Most of his book seems truthful, it contains descriptions of examination papers and advice to parents on which college to choose for their son. Maybe Byron kept the bear temporarily in that turret until it was moved. Some details of Wright and his career will be found in *Mr Hopkins' Men*, by Alex Craik. Mr Hopkins crammed students for the Maths Tripos including many Senior Wranglers. A wrangler is someone who obtains a First in the Maths Tripos and a Senior Wrangler is one who is top of the list in that year. Mr Hopkins' successes included G G Stokes (fluid dynamics) and Arthur Cayley (matrices) who were Senior Wranglers in 1841 and 1842 respectively; also J J Sylvester (geometry and Second Wrangler in 1837). Another was J W Colenso (Second Wrangler in 1836) who became Bishop of Natal. Colenso wrote a jolly book on arithmetic; the 1886 edition has an interesting appendix on decimalization.

Anyway, it seems that Wright broke some regulation (perhaps due to being gored by a bull) that prevented him from taking the Maths Tripos although he was a good mathematician and it was likely he would have been a high wrangler.

One curious loose end is that the turrets in Great Court are not hexagonal at all. They appear to be octagonal but a close inspection from above shows that the internal corner is in fact a right angle so there are only seven sides and thus the turret is an irregular heptagon. (There is a good jigsaw from Wentworth Wooden Puzzles which clearly shows this fact, see www.wentworthpuzzles.com.)

Byron seems to have achieved little academically while in Cambridge but wrote some poetry and generally had a good social life. He did many things that undergraduates do, he suffered being thrown into the fountain which is a fate that befalls many (it's a bit cold but not too deep as I remember).

Regarding pets it is interesting to note that Lady Butler, the Master’s wife around 1970 had a pet. It looked remarkably like a dog and made barking noises much like a dog. Nevertheless, it was classified as a cat and thus permitted!

Years ago, at the assizes, the judge used to stay in A1 in Great Court which is adjacent to the Master's lodge. This is a magnificent room with a glorious bed with silken back embroidered with the letters VR (Queen Victoria slept here). Some years ago it was available as a guest room. Several old chums hired it (in about 1970) for a nostalgic weekend. After a jolly evening they were playing draughts (checkers) by jumping from square to square of the patterned carpet. This disturbed Lady Butler in the Lodge above who came down in her nightgown via a secret door to investigate.

2 Babbage

Charles Babbage was also an undergraduate at Trinity. He came up in 1810 but transferred for some reason to Peterhouse in 1812. Babbage also did not take the Tripos exam for other reasons. In those days, the exam started with a viva with the student sitting on a stool while being asked questions by the Examiner. It is reported that Babbage was unpleasant and maybe blasphemous and was not allowed to sit the examination. He was given an ordinary degree in 1814.

It is conjectured that maybe Babbage deliberately had himself rejected since he did not want to take the exam for fear of being beaten by Herschel from St John's College. Indeed, Herschel was Senior Wrangler in 1813 and the Second Wrangler was Peacock of Trinity. John Herschel was later involved in the discovery of the planet Neptune. George Peacock later became Dean of Ely and supervised the restoration of the cathedral.

Babbage was grumpy concerning the educational state of affairs at the time and formed the Analytical Society with Herschel and Peacock while they were all still undergraduates. This society was eventually instrumental in stimulating modernization. As we know, Babbage was also grumpy with the government many years later regarding the funding of his Analytical Engine.

Babbage was appointed Lucasian professor of mathematics (the same chair that Newton held several centuries earlier) from 1828 to 1839 but never lectured. Incidentally, Newton lived in E staircase in Great Court.

3 Babbage and Brunel

Babbage was a consultant to Brunel during the construction of the Great Western Railway (GWR often known as God’s Wonderful Railway). I gather that he was instrumental in helping Brunel over the matter of the gauge which was set at 7 feet as opposed to the standard gauge of 4 feet and 8½ inches as in Roman chariots and used by most railways. The advantage of the broad gauge is that it permits higher speeds and greater comfort. Moreover, the line from London to Bristol is very level and straight and trains did go like the clappers for the time. Indeed, when the HST diesel trains were introduced on that line in 1975, they were the fastest passenger trains in the world outside Japan and ran at 200kph (125 mph).

Sadly, the force of standardization saw the broad gauge replaced in 1892 by standard gauge throughout. It is interesting to note that the first London Underground railway from Paddington to Farringdon was originally broad gauge and steam hauled from 1863.

As a senior consultant, Babbage was entitled to a company train. These days a senior consultant might have a company car. In those days one might have expected a company horse. But Babbage had rights to a company train!

In 1838, the railway only went from Paddington to Maidenhead. A scary event is described in *Red for Danger* by L T C Rolt and in Vol 1 of *History of the Great Western Railway* by E T MacDermot. Briefly, one Sunday morning Babbage arrives at Paddington and demands his train and is told that nothing else is about so he can use either line. He is just about to set out when Brunel arrives unexpectedly in his own special train that he has taken from Maidenhead. Imagine the disaster if they had met and been on the same line.

The story would have been more dramatic if at night and Babbage was going to Maidenhead to meet a lady at the then notorious Skindles hotel. And we can imagine Brunel having galloped into Maidenhead sweaty from surveying the Sonning cutting and eager to go to his London club for a late dinner. They might have seen each other approaching in the darkness at a closing speed of around 100 mph and prayed they were not on the same track.

Indeed, if they had crashed and Babbage had died that fateful day quite early in his collaboration with Ada Lovelace, then there would have been no analytical engine, it would have been the end of working with Ada Lovelace, our language would not be called Ada, the symposium would not have been held and so you would not be reading this paper.

This incident and others laid the thought of the need for safety on the railways through signalling. Even today, railways are one of the few industries who seem to care about software correctness. Another is avionics.

And now I will turn at last to the matter of events leading to the beautiful Ada language. There were initially two threads of activity, one in Europe and one in the US.

4 LTPL-E

In the mid 1970s, a number of different programming languages were in use in Europe for process control and similar embedded system applications. They included Coral 66 from the UK Ministry of Defence, RTL/2 from Imperial Chemical Industries, LTR in France (RTL backwards), and Pearl in Germany. The European Commission felt that it would be a good idea if the same language could be used throughout Europe and so supported many meetings aimed at defining the basis for a new language. This was perhaps the first stirrings of the objective of ever closer union.

Most meetings were in Brussels and the attendees enjoyed excellent lunches helping to reduce the wine lake and beef mountain which were a problem at the time. The meetings were useful in identifying the requirements for a successor language.

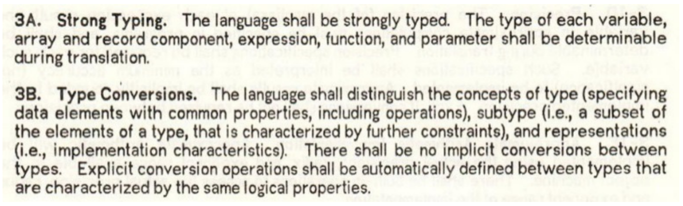
Eventually, the post of chief designer was advertised in the Sunday Times to lead such a development known as LTPL-E (Long Term Procedural Language – Europe). It is said that all those who attended interviews to lead LTPL-E advised that Europe should join with the US in a common development. And indeed the two efforts were merged.

5 The HOL project

In the United States they too thought that there were too many languages. Examples included the Air Force's Jovial (Jules Own Version of the International Algorithmic Language which was based on Algol 58), the Navy's CMS-2 and the Army's Tacpol to name but a few.

Accordingly, the High Order Language project was established under the leadership of Col. William Whitaker. The management team included Philip Wetherall of RSRE (the Royal Signals and Radar Establishment) at Great Malvern in England and David Fisher of IDA (Institute for Defense Analyses) in the US. The first task of the project was to decide what it was all for, that is to define the Requirements. That was an excellent idea – too many projects bash ahead doing something without firmly knowing what it is all for.

So requirements documents emerged and were refined after much deliberation. They were called Strawman, Woodman, Tinman, Ironman, and finally Steelman. A sample of Steelman is shown in Figure 3. Note the relatively abstract level of the requirements. These documents are available at http://iment.com/maida/computer/redref/index.htm.


3A. Strong Typing. The language shall be strongly typed. The type of each variable, array and record component, expression, function, and parameter shall be determinable during translation.

3B. Type Conversions. The language shall distinguish the concepts of type (specifying data elements with common properties, including operations), subtype (i.e., a subset of the elements of a type, that is characterized by further constraints), and representations (i.e., implementation characteristics). There shall be no implicit conversions between types. Explicit conversion operations shall be automatically defined between types that are characterized by the same logical properties.

Figure 3 A sample of Steelman

6 The competition

Four contracts were let for initial designs. They were colour-coded to preserve anonymity to ensure unbiased evaluations. They were as follows:

Green: Honeywell, notionally in Minneapolis but the work was really done at CII-Honeywell-Bull in Versailles. The leader was Jean Ichbiah (now sadly deceased).

Red: Intermetrics in Boston. The leader was Ben Brosgol who has recently retired from AdaCore.

Blue: Softech also in Boston. The leader was John Goodenough who has recently retired (he tells me) after many years at the Software Engineering Institute in Pittsburgh.

Yellow: SRI in Silicon Valley. The leader was Jay Spitzen and he was assisted by many academic consultants.

The four initial drafts are shown in Figure 4. As you can see they are completely anonymous.

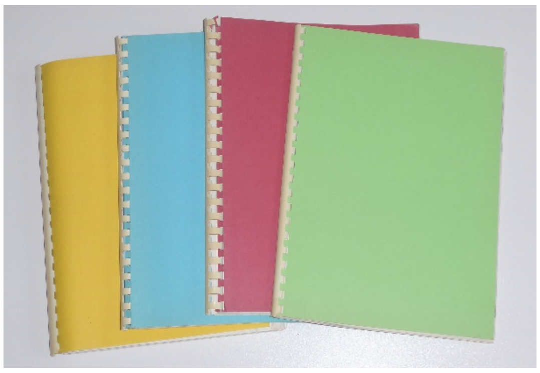


Figure 4. The colourful initial drafts

After one year, Blue and Yellow were eliminated. Blue was interesting but considered somewhat strange. Yellow was rejected largely because it failed to meet the requirements.

Green and Red were then given another year to refine their designs. Red somewhat changed direction and leader and was considered overly ambitious whereas Green consolidated its position.

Green was acclaimed the winner in 1979.

7 The name

The project had gone to plan apart from one vital thing and that was that choosing a name for the new language had not happened.

Eventually, in a wine bar in Paris, a group of management team members chose the name Ada. I understand that the reasons were roughly as follows:

The Pascal language (one of the baselines for Ada) was named after the famous French mathematician Blaise Pascal (1623–1662) well known for his triangle (of binomial coefficients) and his theorem about a hexagon inscribed in a conic. So a good idea to name the new language after a person.

They wanted to honour a woman. Grace Hopper had done much for COBOL.

Ada Lovelace was clearly the world's first programmer. So the name Ada was proposed. Permission was sought from her descendant, the 4th Earl of Lytton. Philip Wetherall from the MoD wrote to the Earl on 10th October 1978. The Earl replied on the 18th to say Yes and observed

that ADA was at the heart of RADAR

But do remember that we always write Ada and not ADA which is the American Dental Association.

The language community were delighted to have Ada as their mascot. Pictures of Ada sprung up on books and documents. And statuettes of Ada continue to be awarded to those making valued contributions to the cause. See Figure 5.

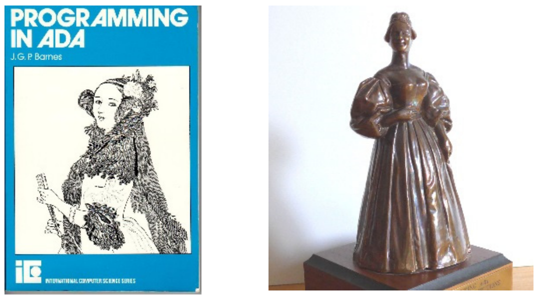


Figure 5. Images of Ada

Many exciting conferences have been held and an especially good one was in London in 1997 when the 5th Earl of Lytton was guest of honour. It was his father that gave permission to use the name Ada.

A particular feature of this conference was entertainment from the New York Village Opera Group. This took the form of a play entitled *The Maiden and the Mandate* which showed the conflict between Lady Ada who wrote excellent software in Ada and the treacherous Senior Hacker who wrote in C. It was based on Trial by Jury by Gilbert and Sullivan. Lady Ada was played by Karen Mason (née Leah). The lovely lady is shown in Figure 6.

Figure 6. Lady Ada

8 The standard

Ada was proclaimed as MIL-STD 1815 on 10th December 1980, the 165th anniversary of the birth of Ada Lovelace in 1815 (note the standard number) at an ACM SIGPLAN conference in Boston. Gosh that was also 40 years ago. After a certain amount of polishing Ada became an ANSI standard in 1983 and an ISO standard in 1987.

For a detailed description of the evolution of the project from requirements to international standard including the correspondence with the 4th Earl of Lytton see the paper *Ada – The Project* by William Whitaker in SIGPLAN Notices.

9 Ada now

We all crave freedom. But freedom takes two forms. There is freedom from problems on the one hand and freedom to do whatever you want on the other. These two freedoms clash. Ada aims to provide freedom from problems by detecting difficulties early in the development of software.

Ada is mainly used for software that matters in areas such as avionics, space, and railways. Many applications are somewhat confidential but I can mention iFACTS, part of the Air Traffic Control system now in use over the London area. There is a demonstration in the computer museum at Bletchley. And I am pleased to say that my daughter Janet is one of the system architects.

iFACTS is written in Ada and SPARK. Most readers will be aware of SPARK which is a subset of Ada with additional contracts (historically called annotations) that is supported by static analysis and proof tools. It has its origins long ago in work done at RSRE in the 1970s by Bob Philips and sponsored by a requirements board chaired by Dame Steve Shirley. Bob Philips later worked with Bernard Carré and they created SPADE which later became SPARK. It is sad to note that both Bob Philips and Bernard Carré are no longer with us.

Ada 2012 incorporates contracts and SPARK 2014 is now integrated into the GNAT Ada toolset. The goal is to show that a program is correct through the use of contracts and formal static proof. Testing can only show the presence of errors and not their absence.

It is pleasing to note that both Dame Steve Shirley and the Earl of Lytton were at the banquet held in Balliol College as part of the symposium celebrations.

Ada is of course still evolving and Ada 202y is imminent. Key enhancements will include further features to enhance correctness and proof and to take further advantage of multiple core processors.

At the present time (with the coronavirus dominating our lives) we are all very much aware of the importance of software to keep us going. One is concerned that much of this software is not written with due diligence. An area of great concern is the automotive industry. I gather that a reputable maker has recently issued a recall of a new model regarding a problem with the brakes. Scary.

10 A sad note

This paper ends on a sad note. Some years ago when at a conference in Paris, a member of the HOL team said he would take me to the elegant wine bar on the Champs Elysées where the name Ada was chosen and we would celebrate with champagne. But ... alas it was now a Burger King!

Acknowledgements

I have consulted many people in preparing this talk and I am very grateful for their help. I fear that I am almost sure to have omitted someone but I must thank (in alphabetical order) Jamie Ayre, Sarah Baldwin, Janet Barnes, Susan Bond, Ben Brosgol, Bill Carlson, Larry Druffel, Terry Froggatt, John Goodenough, Ursula Martin, Karen Mason, Nick Neve, Ed Schonburg, Jonathan Smith, Doron Swade, Tucker Taft, Betty Toole, John Webb.

Bibliography

The following are referenced in the text.

Anon, *The Night Climber's Guide to Trinity*, 3rd edition, Wetherhead, Cambridge, 1960.

Alex D. D. Craik, *Mr Hopkin's Men*, Springer, 2007.

E. T. MacDermot, revised by C. R. Clinker, *History of the Great Western Railway*, Vol 1, revised edition, Ian Allen, 1964.

L. T. C. Rolt, *Red for Danger*, David and Charles, 1966.

William A. Whitaker, "Ada – The Project", in *ACM SIGPLAN Notices*, Vol 28, No 3, March 1993)

John M. F. Wright, *Alma Mater, or Seven Years at the University of Cambridge, by a Trinity Man*, 2 Vols, Black, Young and Young, 1827.

From Physicist to Rocket Scientist and How to Make a CubeSat That Works

Carl Brandon

Vermont Technical College, PO Box 500, Randolph Center, VT 05061; Tel: +1 802 728 9947; email: carl.brandon@vtc.edu

Abstract

With a lengthy background in physics-related computing and involvement with Ada from its beginnings, I had the opportunity to develop a CubeSat using SPARK/Ada. The reliability of this language choice enabled us to make one of the few successful CubeSat missions, the only one using Ada, and the first and still only spacecraft of any kind using SPARK/Ada. We are continuing to add to our CubeSat software.

Keywords: CubeSat, SPARK, Ada.

Introduction

As a child, I became interested in science early, following the Collier's Magazine space series in the 1950s. I built many electronic projects and became particularly interested in physics during my high school course.

I went to Michigan State University in 1962 as a physics major. In 1963 I did my first programming, in hex, on their vacuum tube computer, called MISTIC, that was a copy of ILLIAC 1. It had an electrostatic memory of 1,024 by 40-bit words. My project that summer was designing the extractor coil for the cyclotron.

On the side, I wrote a video game ("Space War") that used its one kilo-pixel CRT. I believe it was the second video game ever written, a few months after one at MIT. A significant part of the design of the first Michigan State cyclotron used this vacuum tube computer. I continued programming for the cyclotron group throughout the rest of my time as an undergraduate, both in assembly and FORTRAN, as we moved to the first Control Data 3600. I was able to see the cyclotron completed and operating by the time I graduated in 1966.

IBM

That summer, I had an internship at the IBM Thomas Watson Research Lab studying the magnetic properties of europium oxide near the Curie point. Returning for the fall, I decided to leave Michigan State during the best physics job market ever (the middle of the Apollo program), resulting in many job offers, and I accepted at the IBM Components Division in East Fishkill, NY. I first designed a high power (60W) PNP transistor to complement an existing NPN design. With two colleagues, we designed IBM's first memory chip, of 128 bits. It was the main memory for the IBM 370. (Figure 1)

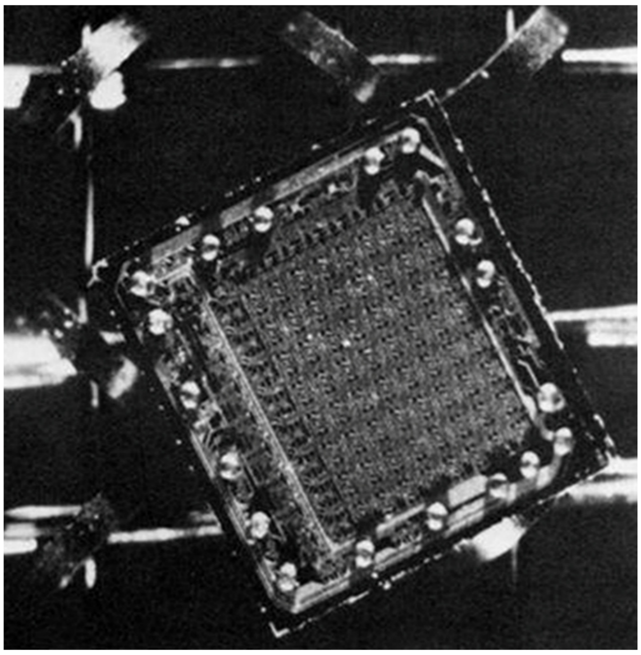


Figure 1. 128 Bit Chip with Core Memory Background

UMass

After two years at IBM, I went to the University of Massachusetts (UMass) for graduate school. I did computer modeling of the seagull (Larus argentatus) soaring flight based on films I made of gulls in flight. For my doctoral research, I filmed the bats (Noctilio albiventris & Tadarida brasiliensis) flying in a wind tunnel. I used a high speed rotating prism camera with a custom strobe I designed and did a frame by frame computer analysis of their flight and related that to their shoulder anatomy.

Vermont Technical College

In 1977 I joined the Science Department at Vermont Tech (part of the Vermont State Colleges) to teach physics, zoology, and anatomy and physiology. Shortly after arriving, I bought and built a Heathkit H-8 computer kit (Figure 2). The math department, which had been teaching BASIC programming on terminals via a remote telephone connected Harris computer, complained that it was so unreliable, they were refusing to use it the next fall. I demonstrated my completed H-8, and Vermont Tech bought four H-8s to replace the four terminals they had. The microcomputer age had begun at the Vermont State Colleges.

Figure 2. Heathkit H-8 Computer

It soon became clear I had more computer background than anyone in Vermont State Colleges. I started teaching programming in BASIC, using H-8 computers. I found a five-day intensive Pascal course in the Boston area by George Poonen [1], which I took, and an Ada course by him the following week. At this time, no compilers were available for Ada. I started teaching Pascal. When the RR Software Janus/Ada compiler became available in 1983, I developed a two-semester Ada sequence, which I taught as part of our new Computer Technology hardware/software degree, which I helped initiate, though not part of that department. In 1986, a new faculty member, Peter Chapin, joined that department, and not knowing Ada, convinced the department to switch to C. Several years later, while working on his Ph.D., he did a project with Ada and "saw the light" [20].

CubeSats

In 2004, I was made aware of CubeSats (10 cm cube, 1 kilogram for a 1U [2]) by a childhood friend who made large satellites about two hours from Vermont Tech. In August 2004, the Vermont NASA Spacegrant Consortium called the Academic Dean and asked her to come to a Technical Advisory Committee meeting, and not being interested, asked the Associate Academic Dean. He was not interested, so asked me (of the 77 faculty, I'm not sure why he asked me, but it turned out to be the right choice), and happy to go to the University of Vermont for a free lunch (this eventually led to a free launch) I went to the meeting. At the end of the session, the director of Spacegrant asked me if I could use $10,000, and I said yes. I sent in a half-page proposal 8 pm Sunday, and 8 am the next morning, he approved it. The grant allowed the purchase of a CubeSat Kit (Figure 3) and some accessories. Since then, I have applied for about 25 NASA grants and received about 35, only not winning three.

Arctic Sea Ice Buoy

While the CubeSat Kit came with essential CubeSat software, which used C, I decided we should find a way to write our CubeSat software in SPARK/Ada. Through our connections at Vermont Spacegrant, we received a contract to build a prototype Alaskan Ice Buoy. The project was to develop a buoy that collects environmental data from the Arctic and transmits that data back to Vermont. The project allowed us to gain experience with CubeSat Kit hardware (we used the CubeSat Kit CPU board), in hopes of using the

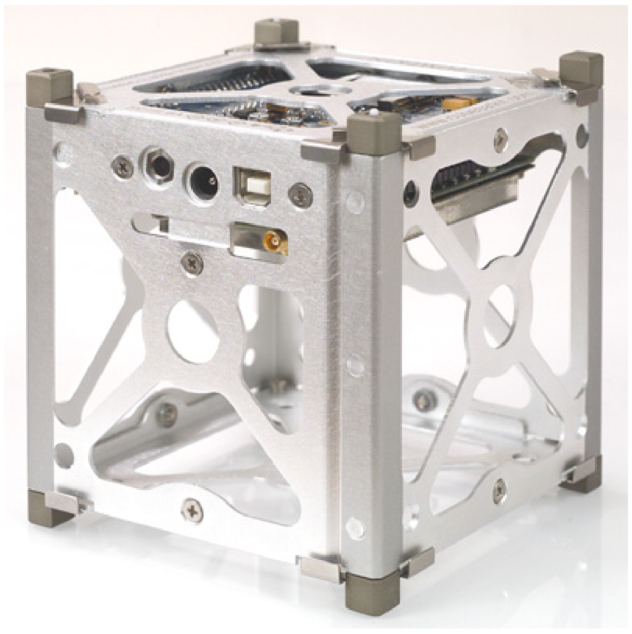


Figure 3. CubeSat Kit Structure & CPU Board

CubeSat Kit platform to launch a satellite into space as a subsequent project. The buoy used the NAL Research Iridium Satellite Modem and GPS. The Iridium data modem sent the position, temperature, wind direction, and speed data to the Iridium network, and then via email attachment to us.

The project was a collaboration with the University of Vermont, which had been studying and mathematically modeling arctic sea ice. The ice forms in the colder months in the northern regions and some melts in the spring and summer months. Unlike terra firma, this ice is continually moving and shifting due to many variables, such as temperature, wind speed, and wind direction. One of the problems that the researchers at the University of Vermont have encountered, having used NASA and ESA radar satellite data, is that they do not have enough data about the sea ice to model it completely. To better model the mechanics of sea ice, they needed to collect data from on the ice itself, which was the goal of the buoy.

The software for the buoy used SPARK, a strict subset of the Ada programming language. We chose it because of its high integrity and reliability. Once we deploy the buoys, there will be no opportunity to correct software bugs or errors, so a reliable program is essential. Unfortunately, the microcontroller in the CubeSat Kit CPU board (Texas Instruments MSP430) has no Ada compiler available. I came up with an idea to get around this and presented it at Ada Europe [3]. This idea was put into practice by a software toolchain created by Peter Chapin (Figure 4). We were able to do this with donated software tools from AdaCore.

In detail, the SPARK code written by the developers was run through the SPARK examiner to ensure that the annotations match the code. The code was then run through SofCheck's AdaMagic compiler, which compiles the Ada code into C code. That C code is then combined with microcontroller device drivers written in C and called from Ada, and it was

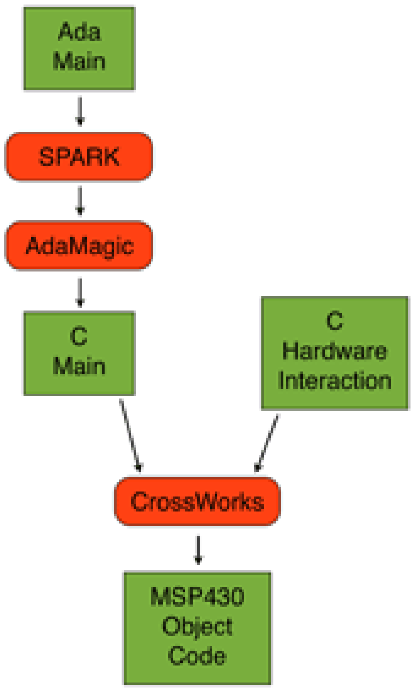


Figure 4. Software Tool Chain

compiled for the MSP430 platform using the Rowley Associates' CrossWorks C compiler.

We completed the prototype, and the plan at the University of Vermont was to deploy a couple of dozen buoys in the Bering Strait, where they had modeled the sea-ice interface. By this time, however, climate change eliminated Bering Strait ice. The second choice, the north coast of Alaska, had ice about 150 km from the shore, and the remote area meant helicopter costs of $5,000 per hour, and they never obtained the follow-up grant. While we missed the fun project of deploying sea ice buoys in the arctic, we learned how we could use SPARK/Ada in our proposed CubeSat.

In July 2009, NASA announced a Consortium Development Competition with a very short two weeks for a letter of intent and then four weeks for the final grant application. Because we had done work with the University of Vermont (100 km) and Norwich University (25 km) and I had been working on concepts for sending a CubeSat to the Moon, we were able to get the letter and grant in early! We received the award with the goal of the project to develop the prototype technologies for a triple CubeSat that would be self-propelled (chemical rockets, or ion drive) from a geosynchronous (communications satellite) launch to the Moon. The chemical rocket would have a lunar lander, and the ion drive spacecraft would go into lunar orbit. The idea for a lunar lander came from a fantasy design from Aalborg University in Denmark (Figure 5). While I realized looking at their image that it was not realistic, I started doing some calculations, and then a detailed design spreadsheet of a chemical propelled lander (Figure 6). With that work already done, we were able to get the grant in on time.

While we had done a fair amount of analysis during the project, in February 2010, NASA announced the Educational Launch of Nanosatellites where NASA would launch a single CubeSat for a cost of $30,000 instead of the commercial launch cost of about $125,000. In the end, they didn’t charge anything. This was a competitive program, and while we weren't finished with the Consortium grant work, an actual CubeSat and launch was too good to pass up. We proposed to test some of our lunar technologies on an actual flight. We were in the first group selected for launch. We were initially scheduled on a SpaceX Falcon 9, in 2012, but that early version of the Falcon on an International Space Station supply mission would not have gone high enough (180 km x 325 km) for more than a few days in orbit. The entire group, now called ELaNa IV chose to wait for a 500 km circular orbit launch the following year. It turned out to be the US Air Force ORS-3 launch on an Orbital Sciences Minotaur 1 rocket (first two stages from a Minuteman II ballistic missile and the third and fourth stages from an Orbital Pegasus air dropped launch vehicle, Figure 7).

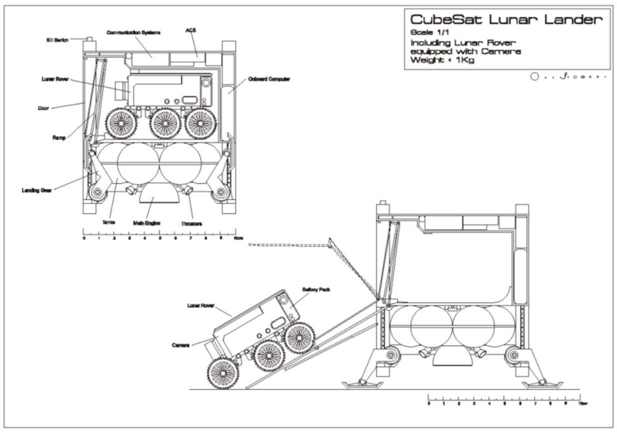


Figure 5. Aalbort University Lunar

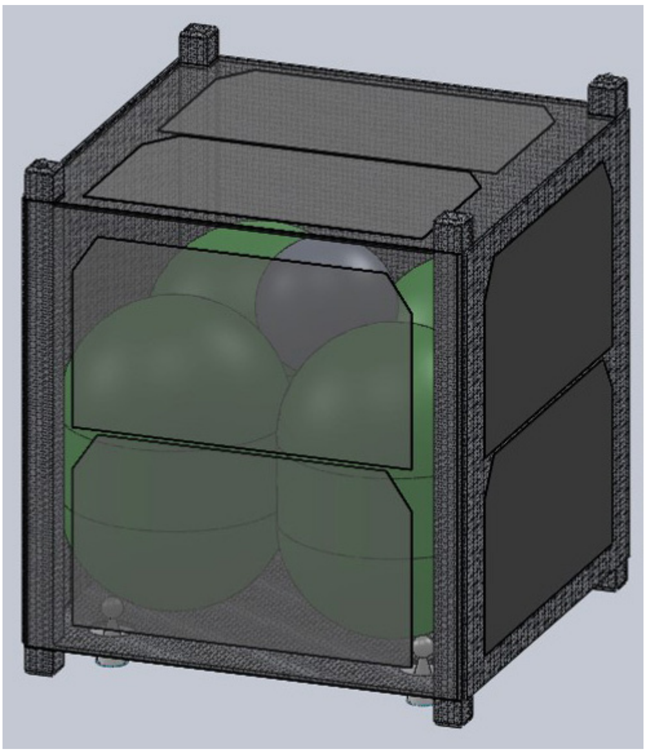


Figure 6. My Lunar Lander Design

Figure 7. Our Launch!

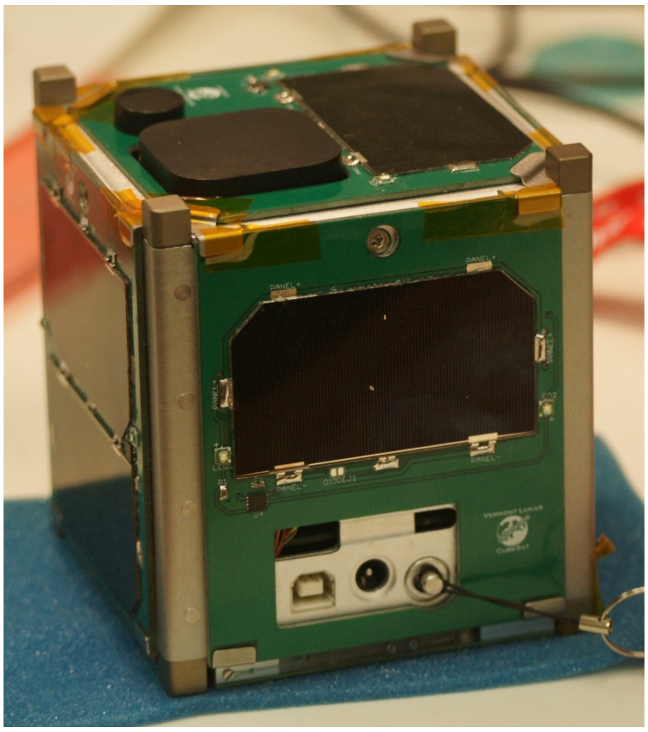


Figure 8. Completed Vermont Lunar CubeSat

Work continued over the next one and one-half years on the hardware design and construction and software design and programming. I did most of the hardware work with assistance from several former Modern Physics students; then, at the company, they formed in Randolph, Vermont, 8 km from Vermont Tech, LED Dynamics. They had space experience having supplied lighting for the Space Shuttle and the International Space Station. Our students designed the software under the supervision of Peter Chapin, with about 80% of the software written by our Software Engineering student Dan Turner [4]. Dan was very considerate, having graduated in May, working on finishing the software all summer so I could deliver the completed CubeSat, called "Vermont Lunar CubeSat." He worked for the student salary of $15 per hour, where he could have made three times that. He quickly got a job near Burlington, Vermont, after the CubeSat was finished and turned down an offer from NSA. By using SPARK/Ada, he was able to complete the software on time. The productivity of 38 LOC per programmer day shown by the Tokeneer project [5] as opposed to the 10-12 LOC per programmer day with C was almost the exact productivity that Dan achieved. The Tokeneer code size was also almost exactly the size of our CubeSat code. Tokeener found four errors in their 10,000 LOC, where a C program would have expected around 1,000 [6]. Without SPARK/Ada, Dan would not have been able to complete the software on time.

On our launch on November 19, 2013, arranged by NASA's ELaNa program on an Air Force rocket contained in addition to 15 classified Air Force satellites (14 3U CubeSats) two NASA 3U (10 cm x 10 cm x 30 cm, 3 kg) CubeSats and twelve university CubeSats of either 1U or 3U size. Since they were classified, we don't know the results of the Air Force CubeSats. Neither NASA CubeSats (both, I believe programmed in C) worked. Of the twelve university CubeSats (all but ours had been programmed in C), eight were never heard from, two had partial contact for about a day (one of which had multiple software errors), one worked for about four months, and ours (Figure 8) worked for two years and two days, 11,071 times around the Earth, 472 million kilometers traveled until it burned up over the mid-Pacific Ocean during re-entry November 21, 2015. We were communicating with it shortly before re-entry.

The primary purpose of our CubeSat was to demonstrate we could make one. We certainly had the smallest group ever to build a satellite. We mostly had three people and, at most, five working on it. For the final three months, just me and Dan Turner, as Peter Chapin was finishing up his Ph.D. We were testing some of the technologies we would use to go to the Moon, primarily the high integrity software system, SPARK/Ada, that would ensure it would work. We had an inertial measurement unit for three-axis linear and rotational accelerations, and magnetic field. We also had a $20 VGA camera from which we received several dozen beautiful photos (Figures 9 & 10). We included the Mad River Glen ski area bumper sticker in miniature (Figure 11). At the ski area bar, there are photos of the bumper sticker all over the world, including at the International Space Station. Ours holds the not likely to be exceeded distance record. Just before launch, we described the control program at another Ada Europe 2013 [7], and further developments at the International Astronautical Congress [8].

Since the Vermont Lunar CubeSat, we have been working on a general-purpose small spacecraft software system called CubedOS [9] designed by Peter Chapin and worked on by our students. We described CubedOS at the High Integrity Language Technology workshop in 2016 [10].

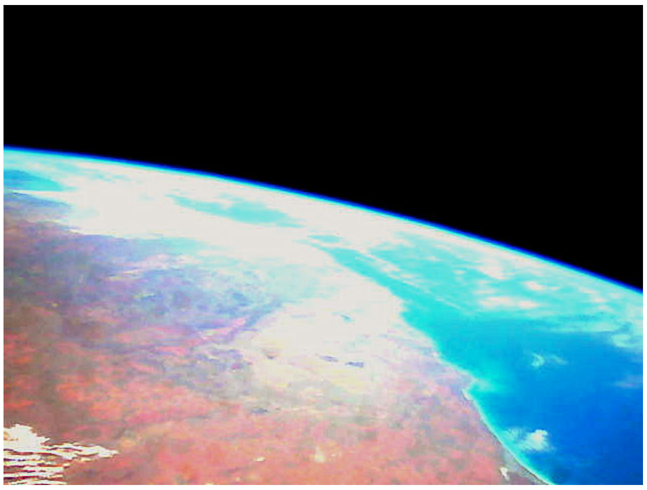


Figure 9. Our First Photo, the North Coast of Western Australia

Figure 10. Photo near Africa

We designed this basic software package so we could expand it for specific spacecraft missions. Our current work is involved in adding additional functionality to CubedOS. In addition to basic other features, we are adding particular modules useful for deep space missions, particularly looking at lunar or Martian missions.

The first added deep space module, Spiral Thrusting, was the implementation of an algorithm developed at NASA's Jet Propulsion Lab (JPL) [11]. Although not designed for CubeSat sized spacecraft, it is uniquely applicable for deep space CubeSats. In more massive spacecraft, angular momentum control (when away from the Earth's magnetic field, where magnetorquers work) in addition to controlling the x-y-axes with the gimbaled main engine, there are auxiliary thrusters for z-axis control. In a CubeSat sized spacecraft (the smallest contemplated for deep space would be a 6U (10 cm x 20 cm x 30 cm, 8 kg) for which there is no room nor mass budget for auxiliary thrusters. Spiral Thrusting allows for 3-axis control with a 2-axis gimbaled thruster. There is a commercially available iodine fuelled ion thruster, the Busek BIT-3 [12]. JPL published this algorithm in 2011, but no one had ever implemented it. Our first Software Engineering Masters student, Chris Farnsworth, implemented it in SPARK/Ada for his Master's project.

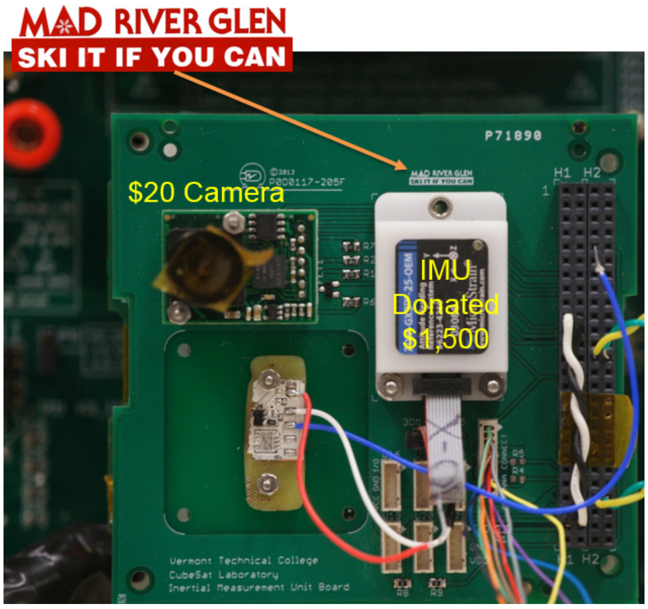


Figure 11. Camera/IMU Board with Mad River Glen Bumper Sticker

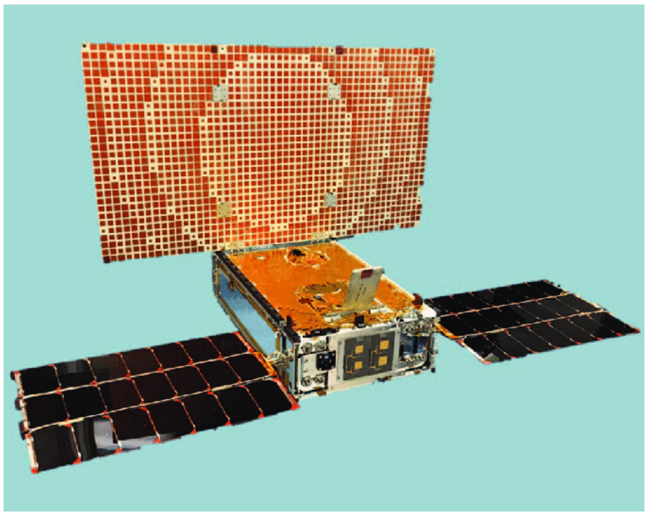


Figure 12. MarCO 6U CubeSat

The module was written with an interface for the Busek BIT-3, but could easily be modified for other thrusters if available. When we demonstrated the software at JPL, they were quite excited to see it implemented and operating. While most of CubedOS is and will be available on Github [13], Spiral Thrusting most likely falls under ITAR and thus is only available to US citizens.

The second module for deep space use is the addition of the JT65 weak signal communication protocol [14]. For most deep-space communications, spacecraft use the NASA Deep Space Network of 70 m and 34 m dishes at Goldstone, CA, Madrid, and Canberra. Besides the high cost, currently $6,000 per hour, it is hard to get time on the very busy DSN system [15]. The JT65 protocol, developed by Joe Taylor, one of my professors in grad school and winner of the 1993 Nobel Prize in physics [16] allows for the accurate reception of radio signals that are as much as 23 dB below the noise floor. Radio amateurs have used JT65 for Moon bounce communication, and I have done this with a student. I was also able to communicate with a station in Japan using JT65

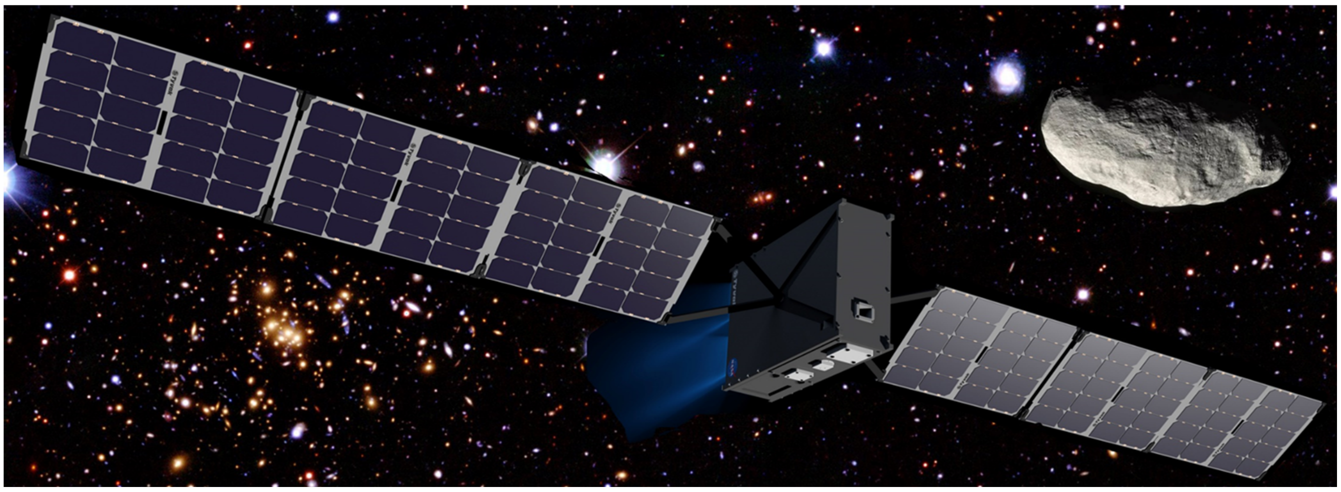


Figure 13. 6U CubeSat with Ion Thruster for a Deep Space mission

with a 1 m pole antenna on my dining room table and 10 W, a distance of 10,600 km. It used to take dozens of Yagi antennas and a couple of kilowatts of power to accomplish Moon bounce. Now it can be done with a one-meter dish and 10 W using JT65.

In calculating the equivalent straight path loss to the Moon bounce loss of 262 dB, it turns out to be equal to the distance from Jupiter to the Earth, when on the same side of the Sun. While the JT65 data rate is very slow at 72 bits per minute, it allows the use of a university ground station for communication. JT65 would be useful for general housekeeping functions and controlling the spacecraft, with three ground stations spaced around the Earth like the DSN, 24 hours per day would yield 13 kB per day and 389 kB per month. So even if the DSN were used occasionally for large chunks of data, the use of JT65 at other times would be beneficial. We currently have two students working this summer on adding JT65 using SPARK/Ada to CubedOS. JT65 requires time coordination between transmitter and receiver. On the Earth and in low Earth orbit (LEO), GPS time signals can be used. When in deep space, out of the range of GPS, another solution is needed. There is a chip-scale atomic clock [17], a rubidium atomic clock with a mass of 35 g and 4.1 cm x 3.5 cm x 1.1 cm dimensions. It has short term stability of about ten ns per hour and less than a one-third second per year drift. This accuracy is more than enough for a multi-year deep space mission. There has already been a deep space mission that tested an antenna on a 6U CubeSat suitable for our purposes. The MarCO CubeSats (Figure 12) used as data relays during the entry descent, and landing of the Insight [18] mission to Mars used a folded reflectarray antenna with a 29 dB gain at 8.4 GHz [19]. While this allowed for a reasonably high data rate to the DSN 70 m dish, it would work just as well for the JT65 low data rate to a 3 m university ground station dish (we have one).

We have a new grant starting in October to add distributed processing to CubedOS. Distributed processing would allow a swarm of CubeSats to work together, sharing computing resources. While continuing to work on CubedOS and adding deep space functionality, we are looking for another satellite mission. We are a small school with about 1,200 students, and a more complicated task than a 1U CubeSat seems desirable. We would have to partner with a larger school or with a NASA center. Our goal would be to supply the software for a deep space mission. Preferably one where we could use our Spiral Thrusting and JT65 modules. If we can do this, we would have created the first and second space missions using SPARK/Ada. While continuing to follow my interest in software supporting my physics applications, I have returned to my childhood interest in space now part of my career. I would not have thought this would happen, but now I can say about what I do, "It IS rocket science"!

References

[1] G. Poonen, “Tutorial on Ada”*, Proceedings of the 1983 annual conference on Computers*, 1983.

[2] Jordi Puig-Suari (Cal Poly) & Bob Twiggs (Stanford U.), *CubeSat Design Specification*, 1999.

[3] Carl Brandon, *“*Use of Ada in a Student CubeSat” Project, *Ada Europe 2008*, Venice, Italy & *Ada User Journal*, Vol 29, No 3, pp 213-216, 2008.

[4] Chad Loseby, Peter Chapin, and Carl Brandon, “Use of SPARK in a Resource-Constrained Embedded System”, *Proceedings of SIGAda 2009*, pp 87-90, 2009.

[5] https://www.adacore.com/tokeneer

[6] Andy German, “Software Static Code Analysis Lessons Learned”, *Crosstalk*, Vol 16, No 11, pp 13-17 (C-130J), 2003.

[7] Carl Brandon and Peter Chapin, *“*SPARK/Ada CubeSat Control Program”, *Proceedings of Ada Europe 2013*, LNCS 7896, pp 51-64, 2013.

[8] Carl Brandon and Peter Chapin, “High Integrity Software for CubeSats and Other Space Missions”*, Proceedings of 66th International Astronautical Congress*, 2015.

[9] http://cubesatlab.org/CubedOS.jsp

[10] Carl Brandon and Peter Chapin, “The Use of SPARK in a Complex Spacecraft”, *Proceedings of the High Integrity Language Technology workshop* (HILT-2016), 2016.

[11] Thomas Randolph, Timothy McElrath, Steven Collins and David Oh, *Three-Axis Electric Propulsion Attitude Control System with a Dual-Axis Gimbaled Thruster*, 47th AIAA/ASME/SAE/ASEE Joint Propulsion Conference 2011, San Diego, CA, 2011.

[12] http://busek.com/index_htm_files/70010819F.pdf

[13] https://github.com/cubesatlab/cubedos

[14] http://physics.princeton.edu/pulsar/K1JT/JT65.pdf

[15] https://eyes.nasa.gov/dsn/dsn.html

[16] https://www.nobelprize.org/prizes/physics/1993/summary/

[17] https://www.microsemi.com/document-portal/doc_download/1243238-space-csac-datasheet

[18] https://mars.nasa.gov/insight/

[19] https://ieeexplore.ieee.org/document/7859458

[20] John McCormick and Peter Chapin, *Building High Integrity Applications with SPARK*, 1st Ed, Cambridge University Press, 2015.

How To Succeed in the Software Business While Giving Away the Source Code: The AdaCore Experience[1]

Benjamin M. Brosgol

AdaCore, 81 Hartwell Ave., Lexington MA 02421, USA; email: brosgol@adacore.com

Abstract

Open-source software, or, more accurately, Freely Licensed Open-Source Software ("FLOSS"), at first appears to present a dilemma when adopted as part of a business model. If users are allowed to access, modify and/or redistribute the source code, how does a software producer protect its intellectual property and sell something that can be easily and legally reproduced?

AdaCore has faced this issue since the company's inception in 1994. Its major commercial product, GNAT Pro Ada, is an Ada development environment based on the GNU Compiler Collection (GCC) from the Free Software Foundation (FSF). AdaCore has implemented an Ada compiler front end and companion run-time libraries and tools and has contributed these components to the FSF. In turn, the GNAT Pro Ada compiler incorporates the GCC back end for a variety of target architectures. By leveraging the GCC back end, AdaCore has made Ada available on a wide range of platforms, both native and embedded, at a significantly reduced effort – indeed, that was the technical rationale for choosing GCC and a design goal of GCC itself. But the challenge of this approach is how to generate a sustained and profitable business. AdaCore's 25 years of FLOSS experience offers an explanation and "lessons learned".

1 Introduction

A software company faces many challenges in realizing its goals for profit and growth, and a fundamental question concerns its business model:

> *Since software is relatively simple to replicate, how does a company protect its investment – i.e., prevent unauthorized uses of its product – without inconveniencing or penalizing customers who use the product legitimately?*

And a related question about the "bottom line":

> *How can a company realize a sufficient revenue stream to be profitable and fund ongoing product enhancements?*

The answers are influenced by the company's choice between proprietary and FLOSS approaches. AdaCore has adopted a FLOSS model, and this article explains the rationale and the company's experience with FLOSS licensing for its major product line, the GNAT Ada development toolset. The article describes the relationship between AdaCore and the FLOSS developer community (principally the FSF), identifies issues that have arisen and explains how they have been addressed, and shows how the FLOSS approach has helped AdaCore sustain a growing and profitable revenue stream over its 25-year history.

2 Early decisions

In the early 1990s a team from New York University was awarded a U.S. government contract to develop a compiler for the language known as *Ada 9X* (later renamed *Ada 95*), a major revision to the original Ada 83 standard. The goal: a user-friendly compiler available to academia at no cost, on multiple platforms. Following discussions with Richard Stallman from the FSF, the project adopted the GCC technology for the compiler back end, with GDB for debugging support, and added an Ada-specific front end and run-time libraries. The resulting toolset was somewhat whimsically named "GNAT", which stood for "GNU NYU Ada Translator". (The acronym expansion has long been abandoned, but the GNAT name has persisted.) Since the GNAT compiler contained GNU software and was intended for teaching and research, the standard GNU General Public License (GPL) was appropriate. This made the GNAT source code available and prevented any of the components from being used in proprietary software.

GNAT was not a production-quality compiler, but the project leaders – Robert Dewar, Edmond Schonberg and Richard Kenner – recognized the commercial potential for a professional Ada 95 development environment and founded AdaCore (then Ada Core Technologies) in 1994 to productize the GNAT technology. Two years later ACT-Europe was founded in France by Cyrille Comar and

[1] This is a condensed version of an article that appeared in *IEEE Software* (Volume 36, Issue 6, Nov.-Dec. 2019), pp. 17-22 (special issue: Twenty Years of Open Source Software). Reprinted with permission.
`https://ieeexplore.ieee.org/document/8792964`

Franco Gasperoni, and the current AdaCore is a result of the subsequent unification of the two companies.

The customer base for the commercial GNAT offering initially comprised two sectors: hardware vendors who needed an Ada 95 compilation system, and software development teams who wanted to use Ada 95. Both sectors' personnel contributed to the technology indirectly, by expressing requirements that would drive product enhancements to be implemented by the AdaCore team.

GPL licensing is appropriate for the compiler but not for the run-time libraries, since customers may need to develop and distribute proprietary or classified applications whose source code has to stay hidden. As explained below, a variation of the GPL license was adopted for the run-time libraries.

3 Why FLOSS?

FLOSS is sometimes touted as being more secure than proprietary software ("given enough eyeballs, all bugs are shallow") but the opposite claim has also been made ("security through obscurity"). In fact, high-profile vulnerabilities have materialized in both types. AdaCore adopted a FLOSS approach not from considerations of software quality but rather for other reasons:

- AdaCore's founders shared the FLOSS philosophy that openly available source code can help the software community advance.
- A production-quality Ada compiler technology targetable to a variety of processors was a major goal from the outset, and the GCC technology offered an effective solution.
- The company founders perceived that making product source code available to customers, and contributing components to the FSF, were not endangering its business. Compiler construction is a specialized field, and the expertise needed to package the FLOSS components into a commercial product (and then to provide the necessary technical support) is high. The risk of the source code being used by potential competitors would be low.

4 Product evolution

From the outset, AdaCore envisioned two user communities for its GNAT technology: individuals or academic institutions adopting Ada for FLOSS software development or teaching, and organizations using Ada to implement professional-grade software for commercial or government-sponsored projects. The version of the GNAT technology for FLOSS developers and academia has helped Ada gain traction in those communities. Known as the "Community Edition," it consists of a no-cost downloadable executable (and source code) for the compiler and accompanying tools / libraries, available on a number of platforms, with licensing appropriate for academic usage and free software development.

The professional version, *GNAT Pro Ada*, shares a common code base with the Community Edition but has several major differences:

- Customers gain access to the toolset through an annual paid subscription, with licensing appropriate for software developers who do not want, or are not allowed, to distribute their source code along with the executable.
- It comes with product support that includes guaranteed rapid response to questions and defect reports, with access to wavefronts (specially generated interim releases) if needed to correct critical problems. Support is provided by the product developers themselves.
- It undergoes a more extensive QA regimen, with nightly regression testing on many platforms.
- Some specialized tools are only available with the GNAT Pro version.

Both the Community Edition and GNAT Pro have at least one major release each year, allowing all users to keep up to date with product enhancements, and, for GNAT Pro, serving as an incentive for customers to renew their subscriptions. The FLOSS approach helps, since the GNAT technology can take advantage of GCC improvements such as new code generators and back end optimizations.

5 AdaCore and the FLOSS Community

AdaCore's has always enjoyed a tightly coupled relationship with the FLOSS community, pulling periodic updates of the GCC and GDB source code from the FSF, and pushing new versions of the Ada-specific components to the FSF. To properly control these interactions, and more generally to manage the release of multiple products on dozens of platforms, AdaCore's development and verification environment has always been highly structured, e.g., with regression tests run at check-in and with extensive QA before product release. The production process is much more like a "cathedral" than a "bazaar"**.**

In addition to contributing its Ada technology to the FSF, AdaCore also makes a variety of tools and libraries available in github repositories where they are open to community enhancements and analysis.

6 Business model and licensing

When AdaCore was founded, contracts from Silicon Graphics and DEC helped fund the initial GNAT development and productization, but these were "one-off" projects. A different source of revenue would be needed to support continued growth and technology enhancements. One approach, common for proprietary software, is to charge a large fee initially and perhaps additional fees for run-time licenses or support. However, this typically results in an uneven (and unpredictable) revenue stream. Moreover, charging for run-time licenses would put Ada at a competitive disadvantage, since other languages do not impose such fees. Instead, AdaCore's business model is based on an annual subscription that reflects the value that the company adds to its FLOSS software: *an expert level of support coupled with assurance about the software's licensing status*.

The subscription model has the benefit of revenue predictability and also incentivizes AdaCore to provide product improvements that encourage customers to renew. There are no run-time license fees.

The GNAT Pro product includes two kinds of software:

- a compiler and companion tools, and
- run-time support libraries.

Different licensing is appropriate for the two cases. The compiler and tools are covered by the GNU General Purpose License, more specifically GPLv3. Under this license, if a user builds the compiler or tool source code (either modified or unchanged), and then distributes a binary version of the result, they also have to make the resulting source code available under the same (GPLv3) terms. This scenario comes up often in practice (for example, Red Hat's distribution of the GNAT Ada toolset from the FSF repository) and helps ensure that FLOSS software stays both freely licensed and open.

AdaCore selected the GPL over other FLOSS licenses because software licensed under the GPL cannot be made part of proprietary software and will always be freely available. In addition, software distributed under GPLv3 cannot be used in hardware that forbids other software from running on that hardware.

The run-time library situation, however, is different. Run-time libraries are linked with object modules from user code, and if the libraries are covered by a GPL license then a user distributing an executable would need to make the source available for their own components. But GNAT Pro is intended for users who may need to develop software that can only be distributed in binary form. The run-time libraries for GNAT Pro (like those for GCC) therefore carry an exception to the GPL:

> *[You] can freely distribute your programs built with the GNAT Pro compiler, including any required library run-time units, using any licensing terms of your choosing.*

This does not apply to the GNAT Community Edition, whose run-time libraries are covered by the GPL without any special exception. Since the Community Edition is intended for free software developers and academic users, the GPL licensing is appropriate.

7 Intellectual property

Although AdaCore is a software product company, it regards its intellectual property ("IP") as being embodied not in its technology *per se* but rather in its development, verification and QA processes / infrastructure. AdaCore's policy of making all of its product source code available with permissive licensing thus helps rather than threatens the company's business: it is in AdaCore's interest if users are familiar with how the technology works, and in fact having the run-time library code accessible and modifiable can be critical in some domains (for example real-time embedded systems). However, although the company's source code is made openly available, its QA tests are not, since many of these tests are proprietary code submitted by AdaCore's customers.

A basic question is how a company can protect its investment – i.e., prevent unauthorized uses of its product – without inconveniencing or penalizing customers who are using the product legitimately. AdaCore addresses this issue in several ways:

- A subscription is based on the number of users, and customers register the personnel who are authorized to download the software, send reports/questions to AdaCore support, etc.
- The product's licensing guarantees are only assured during the duration of the customer's subscription, incentivizing the customer to renew their subscription if they intend to continue using the product.
- There are no run-time license fees, product locks, or other intrusive mechanisms.

8 Issues and "lessons learned"

Coordination among different developers, and integration of components into a coherent product, are challenges for any software development project but are perhaps more acute when FLOSS components are involved. A specific issue for GNAT is when to synchronize with the FSF on incorporating new versions of GCC into the compiler, and, in the other direction, when to commit new versions of the Ada front end, tools and libraries. AdaCore has chosen to be selective with how often it incorporates new versions of GCC, because each upgrade requires significant QA work and often some minor changes to GNAT. On the other hand, the company has pushed its code to the FSF as time has allowed, as long as GCC was in a development stage where changes were permitted.

Another potential issue is how well FLOSS software satisfies customer requirements (reliability, performance, etc.), since, unlike typical projects that start with requirements analysis, FLOSS components are generally developed "bottom up" by contributors who do not necessarily know how the software will be reused. In the case of AdaCore's use of GCC, this has not been a problem: the GCC back end has served well as a retargetable optimizing compiler technology, with sufficient coverage of processors and operating systems to support AdaCore's needs. On the other hand, the standard FSF format for documentation, Texinfo, proved to be somewhat fragile and in general did not provide a modern "look and feel". This issue has been largely ameliorated with the transition to Sphinx. Also, the GCC technology dates back to an era when tools were invoked only through the command line, but modern development teams typically prefer a graphical Integrated Development Environment (IDE). As a result, AdaCore has developed an IDE, originally known as the

GNAT Programming Studio (GPS) and more recently rebranded as GNAT Studio, that has special support for the GNAT technology.

AdaCore's experience as a FLOSS user and contributor have produced several "lessons learned":

- The annual subscription model works well for a customer base that is developing critical software and therefore may need the added assurance that comes with expert and timely support.
- Since support has to be provided for the external FLOSS software (GCC and GDB, in the case of GNAT), the company needs to have internal expertise on these components.
- Synchronization with the FSF must be carefully coordinated (when to upgrade to a new version of GCC or GDB, and when to commit a new version of the Ada components) to avoid disruptions.

In summary, freely licensed open source software can save development cost and be the basis of a profitable business. Its value is not strictly in the product IP but rather in the full complement of processes and the assurances they offer. The key is to provide a high level of support, as well as continued innovation, encouraging customers to continue renewing their subscriptions. AdaCore has traditionally realized a renewal rate of more than 90%, demonstrating that FLOSS can and does work.

ARG Work in Progress IV

Jeff Cousins CEng FIET

Member and former chair of the Ada Rapporteur Group; email: jeffrey.cousins@btinternet.com

Abstract

After a year of the prototyping and validation phase, the finishing touches are being applied to the Ada 202X proposals, though not all features will be implemented in the near future.

1 Introduction

This paper presents a further update on the proposed changes for the next edition of Ada. The previous papers were published in the Vol. 38, No. 1, March 2017, Vol. 39, No. 3, September 2018 and Vol. 40, No.3, September 2019 editions of the AUJ.

As before, Ada Issues (AIs) are first worked on and approved by the Ada Rapporteur Group (ARG).

The ARG follows the following instructions from WG 9, extracted from ISO/IEC JTC 1/SC 22/WG 9 N571:

"The ARG is requested to pay particular attention to the following two categories of improvements:

A. Improvements that will maintain or improve Ada's advantages, especially in those user domains where safety and security are prime concerns;

B. Improvements that will remedy shortcomings in Ada.

Improvements of special interest in these categories are:

- Improving the capabilities of Ada on multi-core and multi-threaded architectures;
- Improving the ability to write and enforce contracts for Ada entities (for instance, via preconditions);
- Improving the use and functionality of the predefined containers;
- Improving support for Unicode in the language and predefined libraries.

These are all examples of improvements in category A, except for the last which is an example of an improvement in category B.

In selecting features for inclusion in the revision, the ARG should consider the following factors:

- Implementability (vendors concerns). Can the proposed feature be implemented at reasonable cost?
- Need (users concerns). Does the proposed feature fulfill (sic) an actual user need?
- Language stability (users concerns). Would the proposed feature appear disturbing to current users?
- Competition and popularity. Does the proposed feature help improve the perception of Ada, and make it more competitive with other languages?
- Interoperability. Does the proposed feature ease problems of interfacing with other languages and systems?
- Language consistency: Is the provision of the feature syntactically and semantically consistent with the language's current structure and design philosophy?"

The proposals from the ARG are then passed to WG 9 (the ISO/IEC Working Group responsible for Ada) for consideration and approval before eventually being consolidated and sent to ISO for formal processing to create a revised international standard.

As you can see, there is a balance to be struck between the first area for improvement being in the area of parallelism, and the first factor for consideration being "Implementability (vendors concerns)". Much feedback from vendors has been incorporated, but the (virtual) WG 9 meeting in June 2020 decided that the parallelism features shall remain in Ada 202X even though there are doubts about whether they will be implemented any time soon.

The first year of prototyping and validation has also been useful for polishing the wording for the new features, so hopefully there won't be many issues remaining for any future corrigendum to correct.

Thanks again to John Barnes for his review comments.

2 WG 9 approved

No proposals were submitted to WG 9 during the first year of prototyping and validation.

3 In the pipeline

These have been approved by the ARG but have yet to be approved by WG 9. With many AIs being ARG approved and entering the "pipeline", but none being WG 9 approved and leaving since October 2018, the pipeline is now bulging even more. The (virtual) WG 9 meeting in June 2020 decided that the submission of ARG-approved AIs to WG 9 should resume, so the backlog should start being cleared.

3.1 From previous "The Future"s

The following proposals to support parallelism were approved by the ARG:

- Global-in and global-out annotations to specify which global objects a subprogram may access, and in which mode (AI12-0079-3). This supersedes the previously approved AI12-0079-1 and *Global aspect and access types used to implement Abstract Data Types (AI12-0240-6)*;
- Default Global aspect for language-defined units (AI12-0302).

With regards to the Real-Time proposals, Atomic and Volatile generic formal types (AI12-0282) was re-opened but after some tweaking, primarily to avoid a backward incompatibility, was subsequently re-approved.

Support for Arithmetic Atomic Operations and Test and Set (AI12-0321) and Add a modular atomic arithmetic package (AI12-0364) provide further Real Time support.

Other proposals that were ARG approved include:

- Making 'Old more sensible (AI12-0280-2);
- Image attributes of language-defined types (AI12-0304);
- Bounded errors associated with procedural iterators (AI12-0326-2);
- Empty function for Container aggregates (AI12-0339);
- Swap for Indefinite_Holders (AI12-0350);
- Add a modular atomic arithmetic package (AI12-0364);
- Changes to Big_Integer and Big_Real (AI12-0366).

Most of the above are expanded below.

Numerous small AIs improved the wording of previously approved AIs without adding new features.

3.2 Global-in and global-out annotations (AI12-0079-3)

These allow the programmer to specify what global data a subprogram uses, in a manner that is similar to that by which subprogram parameters are specified. Specifying the "side effects" (i.e. effects other than via a parameter) of a subprogram makes it easier for static analysis tools to reason. For example:

```
type Operating_Mode_Type is
     (Initialising, Normal, Fallback, Shutting_Down);
type Status_Type is (Success, Inaccurate, Failed);

Data_Table      : …;
Operating_Mode : Operating_Mode_Type;
Status          : Status_Type;

procedure Process_Data_Table
 with
  Global => (in      => Operating_Mode,
             out    => Status,
             in out => Data_Table);
```

This should be fairly familiar to SPARK users, but as we are extending the language proper we may use the reserved words **in**, **out** and **in out** rather than the SPARK terms Input, Output and In_Out.

The Global'Class aspect can be used to specify an upper bound on the set of global variables any subprogram dispatched to may access.

Rather than listing many global variables individually, the reserved word **all** can be used for the set of all global variables, or the reserved word **synchronized** for all synchronized variables (i.e. tasks and protected objects), the implication being that accesses to them are thread-safe.

The intention is that although advanced users may impose stricter requirement on themselves, the typical user should have to specify few, if any, Global aspects. Thus the Global aspect for a library unit usually defaults to Unspecified (sounds a bit like Rumsfeld's "known unknown"!), though to **null** for Pure library units (i.e. no read or write of any global variable). For other entities, the Global aspect defaults to that of the enclosing library unit.

Besides covering any global variables accessed by the body of the subprogram, the Global aspect should also cover those accessed by any preconditions, postconditions, predicates and type invariants. Global variables accessed by other subprograms that the subprogram calls should normally also be identified, though if the other subprogram is passed in as an access-to-subprogram parameter then it is up to the caller of the original subprogram to take account of the effects of whatever subprogram it passes in. If an access-to-variable value is created then presumably the variable that it designates is going to be written, and if an access-to-constant value is created then presumably the constant that it designates is going to be read, so these accesses should be identified too. However, the core language does not check accesses to objects reached via dereferences of access values.

Optional Annex H, for High Integrity Systems, adds restriction No_Unspecified_Globals, disallowing the Global and Global'Class for a library-level entity from being set or defaulting to Unspecified, thereby forcing the specification of Global. It also adds the restriction No_Hidden_Indirect_Globals, requiring that any accesses to objects reached via dereferences of access values are identified.

Annex H also provides an extension for dealing with "handles", for example the File_Type of Text_IO or the Generator of Discrete_Random. Hence, in:

```
procedure Put (File : in File_Type; Item : in String);
```

the File parameter is of mode **in** as the parameter itself isn't modified, yet the state associated with the file is modified. This can now be indicated using an overriding global mode, thus:

```
procedure Put (File : in File_Type; Item : in String)
  with
    Global => overriding in out File;
```

3.3 Making 'Old more sensible (AI12-0280-2)

This was covered in Part III.

3.4 Atomic, Volatile, and Independent generic formal types (AI12-0282)

The aspects Atomic, Volatile, Independent, Atomic_Components, Volatile_Components, and Independent_Components can now be specified for generic formal types. The actual type must have a matching specification, though for backward compatibility reasons the actual types can be Atomic, etc., without the formal types necessarily matching.

3.5 Default Global aspect for language-defined units (AI12-0302)

This was covered in Part III.

3.6 Image attributes of language-defined types (AI12-0304)

Following on from *'Image for all types (AI12-0020)*, it is required that 'Image is required to work for the language defined container types. This uses the new [] array aggregate syntax from *Container aggregates; generalized array aggregates (AI12-0212)*. For Maps it uses the form of a named array aggregate, e.g.:

```
[Key1 => Value1, Key2 => Value2]
```

for Trees the form of a positional array aggregate, e.g.:

```
[[1, 2], [111, 222, 333]]
```

and for null containers the form of a null array aggregate, i.e.:

```
[]
```

3.7 Support for Arithmetic Atomic Operations and Test and Set (AI12-0321)

Following on from *Compare-and-swap for atomic objects (AI12-0234)*, an atomic integer arithmetic package is added, providing operations such as atomic test and set, and atomic increment. This will support the use of locks on multiprocessor platforms.

3.8 Bounded errors associated with procedural iterators (AI12-0326-2)

This was covered in Part III.

3.9 Empty function for Container aggregates (AI12-0339)

Following on from *Array Aggregates; generalized array aggregates (AI12-0212),* the Aggregate aspect now identifies an Empty function, rather than an Empty_<Container> constant, so as to allow a Capacity parameter for those container types that have the notation of capacity (e.g. Vectors).

Thus the example given previously (in Part III) becomes:

```
type Set is tagged private
  with -- Ada 2012 has these
     Constant_Indexing => Constant_Reference,
     Default_Iterator     => Iterate,
     Iterator_Element   => Element_Type,
     …
     Aggregate             => (Empty             => Empty,
                                    Add_Unnamed => Include),
     …
```

3.10 Swap for Indefinite_Holders (AI12-0350)

A Swap operation is added to both Indefinite_Holders and Bounded_Indefinite_Holders. For the former this avoids the overhead of copying the element (and any associated Adjust/Finalize).

```
procedure Swap (Left, Right : in out Holder)
   …
```

3.11 Add a modular atomic arithmetic package (AI12-0364)

Following on from *Support for Arithmetic Atomic Operations and Test and Set (AI12-0321)*, an atomic modular arithmetic package is also added.

4 The Future

4.1 Carried over from previous "The Future"s

Ghost code (AI12-0239), code that is added to support specification and verification, has been voted Hold for a future edition.

Thread-safe Ada libraries (AI12-0139) has been abandoned; no one showed any interest in working on it.

Two alternatives of *Generators/co-routines (AI12-0197)* have been abandoned and two voted Hold for a future edition.

The long-running *Defaults for generic formal parameters (AI12-0205)* has finally been ARG approved. Its spin-off *Defaults for generic formal packages and formal "in out" objects (AI12-0297)* was voted Hold (until a future edition). Of the other AIs relating to making generics easier for the user, *Implicit instantiations (AI12-0215)* has two variants still open, but they may be too difficult, and *Automatic instantiation for generic formal parameters (AI12-0268)* has been abandoned.

No completely new features are under consideration for Ada 202X as we are now approaching its finishing line.

4.2 Floor and other rounding attributes for fixed point types (AI12-0362)

An implementation is permitted to support Floor, Ceiling, and rounding attributes for fixed point types. The AI came in rather late in the day to mandate support, but there is sufficient support to add something.

5 Conclusion

The submission of the Ada 202X proposals to WG 9 is tantalisingly close. Hopefully they will be completed in early 2021; the exact timetable should be decided soon.

ptc® apexada | ptc® objectada®

Complete Ada Solutions for Complex Mission-Critical Systems

- Fast, efficient code generation
- Native or embedded systems deployment
- Support for leading real-time operating systems or bare systems
- Full Ada tasking or deterministic real-time execution

Learn more by visiting: **ptc.com/developer-tools**

ptc

Non-functional Requirements in the ELASTIC Architecture

Luis Nogueira*[*]*, António Barros*[*]*, Cristina Zubia*[**]*, David Faura*[***]*, Daniel Gracia Pérez*[***]*,
***Luis Miguel Pinho*[*]**

[*] *CISTER/ISEP, Polytechnic Institute of Porto, Portugal*
[**] *Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA), Spain*
[***] *Thales Research & Technology, France*

Abstract

The new generation of smart systems require processing a vast amount of information from distributed data sources, while fulfilling non-functional properties related to real-time, energy-efficiency, communication quality and security. The ELASTIC software architecture is being developed to tackle this challenge, considering the complete continuum from the edge to the cloud. This paper provides a brief analysis of the smart application considered in the project, and the requirements emanating from their non-functional properties. The paper then identifies some of the technical constrains imposed to the ELASTIC software architecture to allow guaranteeing the non-functional requirements of the systems.

Keywords: Non-functional requirements, elasticity, computing continuum.

1 Introduction

The ELASTIC project[1] addresses the challenge of extreme-scale analytics, developing a software architecture that incorporates a new elasticity concept, that will enable smart systems to satisfy the performance requirements of extreme-scale analytics workloads. The vision of ELASTIC is that by extending the elasticity concept [2] across the compute continuum in a fog computing environment, combined with the usage of advanced hardware architectures at the edge side, can significantly increase the capabilities of the extreme-scale analytics integrating both responsive data-in-motion and latent data-at-rest analytics into a single solution.

One of the main challenges to be tackled by ELASTIC is the necessity to fulfil the non-functional properties inherited from smart systems, such as real-time, energy efficiency, communication quality or security [1]. In a smart system, such as smart cities, large volumes of data are collected from distributed sensors, transformed, processed and analysed, through a range of hardware and software stages conforming the so-called compute continuum, i.e., from the physical world sensors (commonly referred to as edge computing), to the analytics back-bone in the data-centres (commonly referred to as cloud computing).

This complex and heterogeneous layout presents several challenges, of which an important one refers to non-functional properties inherited from the application domain including real-time, energy-efficiency, quality of communications and security:

- Real-time data analytics is becoming a main pillar in industrial and societal ecosystems. The combination of different data sources and prediction models within real-time control loops, will have an unprecedented impact in domains such as smart cities. Unfortunately, the use of remote cloud technologies makes challenging to provide strict real-time guarantees due to the large and unpredictable communication costs on cloud environments.
- Mobility shows even increased trade-offs and technological difficulties. Mobile devices are largely constrained by the access to energy, as well as suffering from unstable communication, which may increase random communication delays, unstable data throughput, loss of data and temporal unavailability.
- Security is a continuously growing priority for organizations of any size, as it affects data integrity, confidentiality and potentially impacting safety. However, strict security policy management may hinder the communication among services and applications, shrinking overall performance and real-time guarantees.

Overall, while processing time and energy cost of computation is reduced as data analytics is moved to the cloud, the end-to-end communication delay and the performance of the system (in terms of latency) increases and becomes unpredictable, making not possible to derive real-time guarantees. Moreover, as computation is moved to the cloud, the required level of security increases to minimise potential attacks, which may end up affecting the safety assurance levels, hindering the execution and data exchange among edge and cloud resources.

[1] "A Software Architecture for Extreme-Scale Big Data Analytics in Fog Computing Ecosystems", http://www.elastic-project.eu

[2] Elasticity aims at matching the amount of resources allocated to a service with the amount of resources it actually requires, avoiding over- or under-provisioning. Elasticity is a defining characteristic that differentiates cloud computing from previously proposed computing paradigms, such as grid computing.

It is thus necessary that the ELASTIC architecture includes mechanisms which allow the specification of the required level of non-functional properties, the offline analysis of these parameters to determine an appropriate system configuration which enables their fulfilment, and an online monitoring and analysis capability which is able to trigger configuration changes upon detection of level violations. This will be provided via the Non-Functional Requirements (NFR) tool, which will operate both at the offline analysis phase, as well as dynamically during execution.

This paper provides a brief overview of the non-functional system properties which are being considered, as well as some of the technical constrains imposed to the ELASTIC software architecture, both from the requirements emanating from the ELASTIC use cases, as well as from related application domains (smart manufacturing, avionics and automotive). This paper is structured as follows: the next section provides a summary of the considered application domains. The analysis of these domains feed into the non-functional requirements considered, presented in Section 3, and on the integration of the NFR tool in the software architecture, as described in Section 4.

2 Applications with elasticity requirements

The ELASTIC project includes a realistic and challenging smart mobility use-case, but the architecture is not intended to be restricted to this domain. Therefore, an analysis was performed of application requirements emanating from several different domains: Smart Cities/Smart Transportation, Smart Manufacturing, Avionics and Automotive.

2.1 Smart Cities/Smart Transportation

The ELASTIC use-case [2] consists of three different systems, with different types of requirements: a positioning and obstacle detection system; a predictive maintenance and energy consumption system; and an application to manage the coexistence of public and private means of transport.

The first system is a representative set of applications which are intended to increase the safety of the operation of smart transport systems: autonomous localisation of a tram vehicle and obstacle detection to assist the vehicle driver. While traditional solutions adopt sensors along the tracks to detect position, new applications autonomously localise themselves, using sensors (e.g. GPS receivers, accelerometers, RADAR) integrated in a computing system onboard the vehicle. The generated information is also transmitted to other devices connected on the cloud for control systems, which poses new challenges.

At the same time, the current trends to increase the safety of circulating vehicles is to assist the driver with systems that automatically detect and alert for potential hazard situations. Again, systems use specific sensors (e.g. video cameras, RADAR, LIDAR) integrated in a computing system to detect obstacles, determine the potential hazard and alert the driver or even take automatic actions to avoid collision. Information about detected events may be useful on a broader perspective and also transmitted to the cloud.

By onboarding the vehicle with sensors, it is possible also to improve the operation of the full transportation system. Monitoring the rail tracks is a good example of using sensors (e.g. video camera, RADAR, LASER) to acquire significant amounts of data, which must be processed and transmitted to the maintenance management system, to enable predictive maintenance. Monitoring energy consumption is also expected to provide means to potentially improve the electricity energy saving (quite relevant if vehicles rely on battery autonomy in certain catenary-free sectors). This activity may also generate considerable amounts of data that must be processed and delivered so it can be analysed.

Monitoring the interaction between the public and the private transport can potentially increase the safety of both pedestrians and public/private passengers, as traffic can be regulated in real-time, in ways to reduce the risk of accidents. In this case data from multiple different systems is acquired (in a distributed way), and used at the edge level (e.g. to prevent a potential hazard) or at the cloud level (e.g. to adapt the traffic management strategy). Making a coordinated use of the acquired data requires processing, storage and dispatching of traffic data and events. Communication requirements are fundamental to establish where data should be processed.

2.2 Smart Manufacturing

Smart manufacturing is a broad concept. The European Commission DG Communications Networks, Content & Technology defined Smart Manufacturing as real-time workflow application systems assembled from selected data management, modelling, analysis, display, and interface application, in which all information is available when it is needed, where it is needed, and in the form it is most useful, enabling infusion and integration of network based data and information throughout the lifecycle of design, engineering, planning, and production [3]. The development of such smart manufacturing environment depends on the Internet of Things (IoT) capabilities in an industrial context and demands a better integration between IoT and cloud domain. It must be able to handle several data streams from different inputs (such as sensors and actuators, human input, etc.) and be able to store these data into local or distributed cloud infrastructures that are able to communicate effectively with each other.

Smart Manufacturing architectures must be also capable of supporting closed loop control capabilities for low latency automation functionalities. Closed loop control can only achieve high quality results if the underlying infrastructure can provide appropriate Quality of Service (QoS), especially in relation to high timing constraints scenarios. It needs real-time capable physical and transport layers and the use of local clouds or the concept of fog/edge architectures, where the automation is local. Those local clouds provide a protective security fence that protect sensitive automation operations as real-time closed control loops and safety-critical operations.

The concept of elasticity in Smart Manufacturing has not yet been fully explored in the literature. An adaptive elastic implementation of manufacturing process management should consider aspects such as QoS and Service Level Agreement (SLA) metrics and perform an optimization and a runtime adjustment of infrastructural components.

2.3 Avionics

As a reminder, avionics deal about all the electronical systems embedded on an aircraft. With the introduction of the A380 from Airbus or B787 for Boeing, Avionics Systems have changed to a distributed computing architecture, based on Integrated Modular Avionics (IMA) [4]. This computing architecture evolution has maintained the main mandatory avionics properties: the isolation of the avionics applications, by implementing a robust partitioning on the applications being executed on dedicated computing module and on the dedicated deterministic networks. The successful implementation of the robust partitioning has involved the enforcement of the following non-functional properties: the mastery of execution time, the mastery of memory access, the mastery of the access to inputs and outputs, and the mastery of all exchanges. The current avionics architecture is a critical embedded distributed computing platform which is becoming more and more complex. It will be difficult to guarantee that safety and cyber security properties are well maintained during all phases of the flight. So, it will require implementing advanced monitoring mechanisms to confirm in real-time the validity of end-to-end non-functional requirements.

The ITU-T and the actors of the aviation industry have established a working group [5] to evaluate the interest to introduce technologies from the cloud computing and big data analytics domain into the avionic domain and all the others aviation related on-ground domains (maintenance, flight tracking) [6]. The working group has suggested to embed in the aircraft some of the advance monitoring functions performed in the airline maintenance centre as the Flight Data Monitoring [7] and to use the key advantage provided by 'Data in Motion' analytics. The main asset is to implement the fundamental ability to analyse (on the fly, and not analysed after the event occurs), the data collected from various embedded aircraft sources and identify as soon as possible the safety-related and security events that are occurring during the flight to take the most appropriate safety action or to send an alert to the ground station for more investigations and recommendations.

2.4 Automotive

Cars are quickly morphing from an isolated, largely mechanical piece of equipment to one of the most technically sophisticated and connected platforms on the planet. From entertainment and navigation to driver assistance, crash avoidance, and autopilot, today’s cars are vastly different from those of a few years ago.

One of the main requirements of these new functionalities is the need for cars to sense and understand the environment, for which numerous sensors are being used, including cameras, radar, and LIDAR [8], generating huge amounts of data. Using multiple sensor technologies also improves the safety level of the car and at the same time can relax the safety requirement for each individual sensor.

At the same time, cars will interact with each other and have access to each other’s information or information about their surroundings [9]. Future automotive safety applications based on vehicle-to-vehicle and vehicle-to-infrastructure communication are regarded as a means for decreasing the number of fatal traffic accidents.

Developing and deploying autonomous driving systems requires the ability to collect, store and manage massive amounts of data, high performance computing capacity and advanced deep learning frameworks, along the capability to do real-time processing of local rules and events in the vehicle. Therefore, it is also important to consider the cloud infrastructure [10], which the potential to enable a wide range of applications and new paradigms in autonomous vehicles, overcoming the limitations posed by standalone implementations, but at the same time opens challenges related to real-time response, communication quality and security.

Cybersecurity is thus an important concern as it relates to both trust and acceptance of autonomous driving technology as well as safety. Like all other connected communications networks, concerns remain that hackers could steal personal information and spy on people or that malicious control of a vehicle could cause personal harm or disrupt traffic flow [11][12].

Sufficient power needs to be available to operate all these car components at all times needed, which of course involves when the car is in motion, but also when it is idle, and even to some degree when the car is no longer considered in active use for a driving journey. It is taken for granted that the autonomous systems being tested right now require a lot of computing power, but it is easy to overlook that all of that computing power comes at a cost of actual electric power. Much more complex autonomous systems will also require a lot more computing power to run, which translates into energy consumption.

3 Non-functional requirements for the ELASTIC software architecture

Based on the analysis of the considered application domains, the following section provides a summary of the requirements (67 in total [1]) which are placed to the ELASTIC software architecture.

3.1 Timing requirements

Coping with real-time computing across the compute continuum requires the ability to specify and manage different timing perspectives. These systems have either local (i.e. at the edge node) or end-to-end (i.e. edge-to-cloud) requirements that define the boundaries of their useful operation. Local timing requirements can be critical (e.g. to timely brake the tram vehicle in order to avoid a collision), but also can be end-to-end timing requirements (e.g. informing the control centre about the current localisation of the tram vehicle). Two main challenges arise: tasks deployed at the edge (for example, on board the connected car) need to guarantee “hard real-time” responses (e.g. very low latency) and those deployed at the cloud need to guarantee certain QoS levels regarding time: right-time or “soft real-time” guarantees.

Closer to the environment, at the edge, tight timing mapping and scheduling approaches can be used, while at the cloud, time is measured in terms of average statistical performance with Quality of Service (QoS) constraints. These perspectives complement each other, and ELASTIC will provide solutions that will allow to dynamically deploy application components distributed over the system, to provide the required response time to applications, whilst optimizing energy and communication costs. ELASTIC will provide analysis and tools to determine offline a predictive response-time for a particular application, and then dynamically adjusting the system resources to which the application is mapped, depending on the actual load of the system. A particular concern will be given to model the different time scales of the system, providing execution models that consider temporal behaviour of applications from the fast interactions at the edge side to the wider timing perspective of the cloud side.

3.2 Energy requirements

The energy requirements placed by mobile and autonomous systems require that special attention is given to energy efficiency and trade-offs between energy consumption and other non-functional properties (current prototypes for fully autonomous driving systems consume the equivalent energy of 50 to 100 laptops [13]).

This requires that systems are augmented with “introspection” capabilities in terms of power consumption, and the use of energy-aware execution models, from the hardware platform to the holistic system. This will allow to propagate workload-specific information from the run-time to the decision-making module, which can be exploited to better adapt to the requirements, as well as to the time predictability and security optimisation levels. Furthermore, a richer knowledge of applications’ requirements and concurrency structure, coupled with precise energy models for the underlying hardware, combined with the possibility of dynamically switching between edge and cloud deployments, constitutes an enabling factor towards larger energy savings, and the development of novel online and offline optimization strategies. For this purpose, it is required to devise methods to extract information on workload specifications, including non-functional specifications, and platform characteristics, impacting on energy efficiency. This will enable allocation strategies and run-time mechanisms that consider energy information and energy-aware execution models to efficiency tune power consumption over the complete continuum.

3.3 Communication requirements

The main objective of the communication in the ELASTIC use cases [2] is to transfer information between applications from the edge side to applications hosted in a private cloud. The data exchange along the ELASTIC fog computing architecture is mainly done through wireless interfaces (LTE and Wi-Fi).

The compute continuum needed to implement safety related communication channel [14] to allow the communication between the IT environment (clouds) and the Critical Edge Environment (Tram, Aircraft, Car,...). This communication architecture must be able to transfer monitoring data, images and video from the edge nodes to the cloud and the control data from the cloud to the edge nodes and providing at the same time safe end-to-end transfer and end-to-end security.

ELASTIC must ensure reliable communication channels to avoid information loss and must implement verification mechanisms for data communication, in order to realise reliable communication on top of communication infrastructures with unknown quality (mechanisms for out-of-order-delivery and acknowledgement, for example). It must be shown that the end-to-end information being transmitted is complete, not altered, not missing, on time and must be monitored at runtime. All this potential monitoring function must be distributed between each node of the compute continuum and the Non-Functional Requirements (NFR) tool.

The compute continuum architecture by design uses different kinds of network protocols and standards depending mainly on the type of the network physical medium. This communication architecture must also implement a safe full-duplex communication system to forward the information. So, low latency communication protocols must be supported by ELASTIC in order to collect real-time data from devices and support real-time traffic to process sensor data.

3.4 Security requirements

The security requirements of the different use cases must be considered in the ELASTIC architecture. Depending on the trustworthy level of the application domain (railway, avionics, automotive,…) and the risks associated to each device (edge, network, cloud, …) in the architecture, some key properties and the embedded associated mechanisms are needed to be implemented in order to master the behaviour of the communication system and to ensure the end-to-end enforcement of the security properties defined during the design phase.

The ELASTIC communication architecture should implement secure bidirectional data flow channels in order to provide privacy, data integrity and authentication between the different actors present in the architecture. It is recommended to identify at design time the critical dataflow involved in the communication Use Cases and to define their main characteristics. The ELASTIC wired and wireless communication system must ensure for the most critical dataflow a robust partitioning [15] of its allocated bandwidth to prevent Denial of Service attack. The critical dataflow bandwidth availability and their main metadata fields (@Source, @Dest, Protocol, Timestamp, Geolocation, CRC …) must be monitored at runtime in each communication nodes and each erroneous message must be deleted. The availability and metadata field errors must be reported to the NFR Tool communication supervisors function.

ELASTIC shall also satisfy GDPR [16] for managing personal data. There are different critical stages where personal information is managed. Some examples include the facility of data erasure with no restrictions, the creation of a hierarchy of roles and users for controlling the access to restricted components of the system, etc. A RBAC (rule-based access control) can be implemented in order to control the access of each device to certain endpoints of services. The use of device certificates and mutual authentication would offer a higher level of security, and improved control of the connected devices and permissions.

3.5 Requirements interdependency

Interdependency between the requirements provides further challenges to the architecture. The expected large amounts of data generated by new applications raises the relevance of timing and communication requirements. Processing data on the edge allows for tighter timing properties, also potentially reducing the amount of data that has to be delivered to the cloud. However, the cloud has more computing power to perform data analysis, and potentially larger energy envelops, but the end-to-end communication delay and the latency of the system increases and becomes unpredictable.

Also, as computation is moved to the cloud, the required level of security increases to minimise potential attacks, which may end up affecting the safety assurance levels, hindering the execution and data exchange among edge and cloud resources.

4 Integrating Non-Functional Requirements in the ELASTIC software architecture

Contemporary cloud computing solutions, both research projects and commercial products, have mainly focused on providing functionalities at levels close to the infrastructure. Furthermore, they tend to focus on functional aspects only.

In order to provide an improved ecosystem, which considers the full compute continuum, there is a great need for analysis and monitoring tools that support higher-level concerns and non-functional aspects in a comprehensive manner, from the edge to the cloud. Therefore, the NFR tool of the ELASTIC architecture will operate both at the analysis phase, and during execution.

4.1 Analysis

The goal of the analysis phase is to guarantee the fulfilment of the system non-functional properties, considering the potential trade-offs between performance, predictability, energy-efficiency, communication quality and security. The result of this analysis is a set of possible initial deployment configurations. For example, in one configuration, to fulfil security requirements, a stronger encryption algorithm might be used than another alternative service configuration. However, using a stronger encryption algorithm may lead to consuming more memory or processing capacity and CPU time, and, in this way, it impacts memory and performance requirements. Today, a plethora of verification techniques exist for different layers in the cloud stack (IaaS, PaaS, SaaS) that typically address a single non-functional property.

The analysis phase should be able to carefully identify how satisfying and fulfilling one requirement can impair the satisfaction of other requirements in the system. Establishing and maintaining such interdependencies during the development process and the lifecycle of the system is also an important point, taking into account the evolution of the software architecture and the introduction of new requirements or the modification of existing ones. Moreover, not only requirements can have impact on each other, but also one requirement usually crosscuts different parts of a system. For example, achieving security in a system requires design decisions for different parts of a system spanning from user interfaces (e.g., what a user can enter as input), database backends, communication protocols, network topology, and so on.

4.2 Online monitor

Deployment decisions should be made in light of a target system, aiming for high quality of the system deployed under given constraints. However, in order to support deployment decisions, it is essential to identify concrete measures as a basis for decision making and evaluation of the proposed solutions. Such measures can be static, as described in the previous section, and dynamic, in the case of systems that evolve continuously as the workloads, allocated resources and requirements of these systems change over time.

Therefore, runtime monitoring of requirements should be used to guide this evolution towards configurations that are guaranteed to satisfy the system's overall requirements. Monitoring should be used to identify the scenario the system operates in, and to select a model whose quantitative verification enables the detection or, sometimes, prediction of violations. The subsequent synthesis and execution of a provably correct reconfiguration plan help the system to re-instate or maintain compliance with the expected level of service.

The monitoring phase also requires that the ELASTIC software architecture is able to provide information on the resource usage and application execution in the nodes, and dynamically re-map and schedule components considering the execution profile identified by the monitor.

5 Conclusions

The ELASTIC project targets the development of a software architecture that incorporates a new elasticity concept, which will enable smart systems to satisfy the performance requirements of extreme-scale analytics workloads. A main challenge of the project is the need to consider the non-functional properties inherited from smart systems, such as real-time, energy efficiency, communication quality or security. This paper provides a summary of the non-functional properties identified in the scope of the ELASTIC project, and the technical requirements imposed into the ELASTIC software architecture.

Acknowledgements

This work has been financially supported by the European commission through the ELASTIC project (H2020 grant agreement 825473).

References

[1] ELASTIC Consortium, Deliverable D4.2: "Non-functional properties analysis and constraints specification", May 2019

[2] ELASTIC Consortium, Deliverable D1.1: "Use case requirement specification and definition", May 2019

[3] European Commission DG Communications Networks, Content & Technology, "Definition of a Research and Innovation Policy Leveraging Cloud Computing and IoT Combination", ISBN 978-92-79-47760-7, 2014.

[4] Henning Butz, "Open integrated modular avionic (IMA): State of the art and future development road", Technical report, Department of Avionic Systems, Airbus Deutschland GmbH, 2010.

[5] ITU-T, Focus Group on Aviation Applications of Cloud Computing for Flight Data Monitoring.

[6] ITU-T, "Existing and emerging technologies of cloud computing and data analytics", 2016

[7] EASA, "Flight Data Monitoring on ATR Aircraft", 2016

[8] Rajeev Thakur, "Infrared Sensors for Autonomous Vehicles", *Recent Development in Optoelectronic Devices*, IntechOpen, 2017. DOI: 10.5772/ intechopen.70577

[9] M. Campbell, M. Egerstedt, J.P. How, R.M. Murray, "Autonomous driving in urban environments: approaches, lessons and challenges", *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 368:4649-4672, 2010. DOI: 10.1098/rsta.2010.0110

[10] S. Liu, J. Tang, C. Wang, Q. Wang, J. L. Gaudiot, "A Unified Cloud Platform for Autonomous Driving", *Computer*, 50: 42-49, 2017.

[11] Andy Greenberg, "The Jeep Hackers are Back to Prove Car Hacking Can Get Much Worse", *Wired*, 2018. Available at: https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/

[12] Jared Gall, "Can a Connected Car Ever Be Safe from hacking?", *Car and Driver*, 2017. Available at: https://www.caranddriver.com/features/a15079914/ can-a-connected-car-ever-be-safe-from-hacking-feature/

[13] Gabrielle Coppola, Esha Dey, "Driverless Cars Are Giving Engineers a Fuel Economy Headache", *Bloomberg*, 2017. Available at: https://www.bloomberg.com/news/articles/ 2017-10-11/driverless-cars-are-giving-engineers-a-fuel-economy-headache

[14] CENELEC, "EN 50159: Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems", 2011.

[15] P.Toillon, P.B.Champeaux, D.Faura, W.Terroy, M.Gatti, "An optimized answer toward a Switchless Avionics Communication Network", *DASC*, 2015.

[16] EU General Data Protection Regulation, https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en.

Forty Years On and Going Strong

John Barnes

11 Albert Road, Caversham, Reading, RG4 7AN, UK; Tel: +44 118 9474125; email: john@jbinformatics.co.uk

Hello readers:

It is forty years of the Ada User Journal. It is ten years since it was thirty years ago and I wrote a short paper giving the history of the journal and how it had evolved. In this current issue you will also find a revamp of a paper about the evolution of the language and some mutterings about Lord Byron and his bear.

This has made me contemplate some highlights of the past 40 years especially with regard to memorable conferences.

But first some remarks about a book that I found in a second-hand bookshop in Henley-on-Thames in around 1997. It was Mathematical Bafflers by Angela Dunn. What really intrigued me was that tucked into the front was the shipping packaging slip. This showed that it was purchased by David C Sullivan of Litton Industries in Beverly Hills. The date of purchase was 27 June 1967, price (including Calif state tax) was $7.45. It contains puzzles from an internal weekly magazine of Litton Industries. The foreword tells us that these have been submitted by readers and answers are included. Two questions struck, why was this book thousands of miles away from California? And who was Miss Dunn? Was she now a maths professor at a college in California?

At a SigAda conference in Washington in 1998 I met some people from Litton but they could tell me little. However, at the SigAda conference in Redondo Beach in 1999, I obtained Miss Dunn's phone number from a local library. I called her and we met for dinner in a Mexican joint in Beverly Hills. It turned out that the editorship had been a transitory part of her life and she had spent most of her career interviewing film stars and writing pieces for magazines. She gave me a list of stars whom she had met. It starts with Fred Astaire and Bob Hope! But the truly interesting thing was that her middle name was Fox. She was the granddaughter of the founder of Twentieth Century Fox. Amazing!

Two Ada-Europe conferences stand out in my memory. One was awful, one was wonderful. The awful one was in Frankfurt in 1995. The first disaster was that I broke my spectacles when changing trains in Brussels. So I was fairly blind for the week. The other bad thing was that it was a joint conference with Eurospace, they gave an opening address roughly saying, we are not that keen on Ada!

The following conference in 1996 organized by Alfred Strohmeier was in Montreux. It was wonderful! Alfred had persuaded the manager of the hotel to let the executive suite to the President for the price of a normal room on grounds that he could then let an additional normal room. The suite included a private bar, a meeting room, a secretaries room, a veranda with seating for 24. The meeting room was superb for our board meetings. I even gave tours of my rooms. And one could control the fancy B&O sound system lying in the bath by foot-controlled switches!

At the gorgeous Ada-Europe conference in Venice in 2008, I was given honorary membership of Ada-Europe (I had dragged on as President for some years but was relieved to hand over to Erhard Ploedereder). In exchange I was asked to give a smallish speech/address at future conferences (much to the dismay of my wife). These addresses have continued and generally comprise a few remarks about the progress of Ada, probably a wretched joke, and ending with some sort of mathematical puzzle.

Since many of us are locked down by the virus, I thought maybe a puzzle or two would not come amiss right now.

One of the puzzles was the sum

```
  CROSS
  ROADS
-------------
DANGER
```

where the letters represent digits. The answer is

96233 + 62513 = 158746

Another particularly nasty one was

```
      *  *  *
      *  *  *
-----------------
*  *  *
   *  *  *
      *  *  *
-----------------
*  *  *  *  *
```

where we are told that each of the ten digits from 0 to 9 occurs exactly twice in this multiplication puzzle.

Other puzzles of a similar sort are

HOCUS + POCUS = PRESTO

COUPLE + COUPLE = QUARTET

ZEROES + ONES = BINARY

FISH + N + CHIPS = SUPPER

and using multiplication for a change

PI * R * R = AREA

The answers might be given in the next issue of the Journal.

Take care.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o KU Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Hubert B. Keller
Karlsruher Institut für Technologie (KIT)
Institut für Angewandte Informatik (IAI)
Campus Nord, Gebäude 445, Raum 243
Postfach 3640
76021 Karlsruhe
Germany
Email: Hubert.Keller@kit.edu
URL: ada-deutschland.de

Ada-France

attn: J-P Rosen
115, avenue du Maine
75014 Paris
France
URL: www.ada-france.org

Ada-Spain

attn. Sergio Sáez
DISCA-ETSINF-Edificio 1G
Universitat Politècnica de València
Camino de Vera s/n
E46022 Valencia
Spain
Phone: +34-963-877-007, Ext. 75741
Email: ssaez@disca.upv.es
URL: www.adaspain.org

Ada-Switzerland

c/o Ahlan Marriott
Altweg 5
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: president@ada-switzerland.ch
URL: www.ada-switzerland.ch