ADA USER JOURNAL

Volume 36
Number 1
March 2015

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.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

The year 2015 is for sure a special year, with the celebration of the 200th anniversary of Ada Lovelace, the 19th century mathematician from whom the language takes the name. In this special year, it is important that we look back to analyze how much we know about Ada's life and work. Ada undoubtedly deserves a place in the computing history, being a model for all of us (whatever the gender), both due to her achievements as well as the difficulties she faced. In that context, the Ada User Journal will join the celebrations, and will feature during the year, articles on Ada the person, in parallel with Ada, the language. We hope that the reader will both enjoy and learn a little more (on both subjects).

The year thus starts with an article by Suw Charman-Anderson, journalist and social technologist, and the founder of Ada Lovelace Day. In the article, "Ada Lovelace: Victorian computing visionary", Suw gives a perspective on Ada's life and achievements, and presents why Ada is the perfect person to be the model for women in technology. I would like to acknowledge Suw for the work on clarifying many of the misconceptions and disputes on Ada's work.

I would like also to note our readers to stay tuned to the forthcoming Ada-Europe conference. The 20th International Conference on Reliable Software Technologies – Ada-Europe 2015, will take place 22-26 of June 2015 in Madrid, Spain. The advance program of the conference, which can be found in the forthcoming events section of the issue, illustrates that it will be a remarkable event, due to its rich program.

On the program of the conference, I would like to highlight the sessions of technical papers and industrial presentations, as well as the special featured keynote talks by Jon Pérez, on certification of mixed-criticality systems based on multicore and partitioning in EC-61508, by Javier Rodríguez on Software Development of Safety-Critical Railway Systems and Andras Balazs on the on-board computer of the Philae lander in the context of the Rosetta space mission.

The conference week will also encompass nine tutorials (with topics including parallel programming, memory management, coding standards, timing analysis, Ada 2012 in practice, Ada and Python, software measures and design concepts and real-time and embedded programming), and two workshops on "Challenges and new Approaches for Dependable and Cyber-Physical Systems Engineering" and "Architecture Centric Virtual Integration". A special note, on this special year, to the Steering Committee of the first workshop, which intends to promote women engaged in science, in both industry and academia.

The program of the week also includes space for networking and interaction, both during the day as well as in the featured social events. And, closing the circle, for sure Ada Lovelace will be remembered during the week.

The technical part of this issue of the Journal also provides a rich set of contents. It provides an article by Jacob Sparre-Andersen, from JSA Research and Innovation, Denmark, on Implementing Ada 2012 Persistent Containers with Memory-Mapping, and a paper by a group of authors from Universidad Politécnica de Cartagena, Spain, which presents a task-based concurrency scheme to support component–based applications with real–time requirements

Luís Miguel Pinho
Porto
March 2015
Email: AUJ_Editor@Ada-Europe.org

Quarterly News Digest

Jacob Sparre Andersen

Jacob Sparre Andersen Research & Innovation. Email: jacob@jacob-sparre.dk

Contents

Ada-related Events

[To give an idea about the many Ada-related events organised by local groups, some information is included here. If you are organising such an event feel free to inform us as soon as possible. If you attended one please consider writing a small report for the Ada User Journal. —sparre]

Ada-Europe 2015: Call for Papers

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Mon, 24 Nov 2014 23:37:29 +0000
Subject: 2nd CfP Ada-Europe 2015 Conference, Madrid, Spain
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

2nd Call for Papers

20th International Conference on Reliable Software Technologies - Ada-Europe 2015

22-26 June 2015, Madrid, Spain

http://www.ada-europe.org/conference2015

Organised by Ada-Spain on behalf of Ada-Europe, in cooperation with ACM SIGAda, SIGBED, SIGPLAN and the Ada Resource Association (ARA)

*** DEADLINE 11 JANUARY 2015 ***

*** Web submission site open ***

Ada-Europe organises annual international conferences since the early 80's. This is the 20th 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), and Paris, France ('14).

General Information

The 20th International Conference on Reliable Software Technologies - Ada-Europe 2015 will take place in Madrid, Spain. Following its traditional style, the conference will span a full week, including a three-day technical program and vendor exhibition from Tuesday to Thursday, along with parallel tutorials and workshops on Monday and Friday.

Schedule

11 January 2015: Submission of regular papers, tutorial and workshop proposals

25 January 2015: Submission of industrial presentation proposals

1 March 2015: Notification of acceptance to all authors

29 March 2015: Camera-ready version of regular papers required

12 April 2015: Industrial presentation abstracts required

17 May 2015: Tutorial and workshop materials required

Topics

The conference has over the years become 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 representing industry, academia and government organisations active in the promotion and development of reliable software technologies.

Topics of interest to this edition of the conference include but are not limited to:

- Multicore and Manycore Programming: Predictable Programming Approaches for Multicore and Manycore Systems, Parallel Programming Models, Scheduling Analysis Techniques.
- Real-Time and Embedded Systems: Real-Time Scheduling, Design Methods and Techniques, Architecture Modelling, HW/SW Co-Design, Reliability and Performance Analysis.
- 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: Design Patterns, Frameworks, Architecture- Centered Development, Component-based Design and Development.
- Methods and Techniques for 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.
- Software Quality: Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems.
- Mainstream and Emerging Applications: Manufacturing, Robotics, Avionics, Space, Health Care, Transportation, Cloud Environments, Smart Energy systems, Serious Games, etc.
- Experience Reports in Reliable System Development: Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics.
- Experiences with Ada and its Future: Reviews of the Ada 2012 new 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 Regular Papers

Authors of regular papers which are to undergo peer review for acceptance are invited to submit original contributions. Paper submissions shall not exceed 14 LNCS-style pages in length. Authors shall submit their work via EasyChair following the relevant link https://easychair.org/conferences/?conf=adaeurope2015. The format for submission is solely PDF.

Proceedings

The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer, and will be available at the start of the conference. The authors of accepted regular papers shall prepare camera-ready submissions in full conformance with the LNCS style, not exceeding 14 pages and strictly by March 29, 2015. For format and style guidelines authors should refer to http://www.springer.de/comp/lncs/authors.html.

Failure to comply and to register for the conference by that date will prevent the paper from appearing in the proceedings.

The CiteSeerX Venue Impact Factor has the Conference in the top quarter. Microsoft Academic Search has it in the top third for conferences on programming languages by number of citations in the last 10 years. The conference is listed in DBLP, SCOPUS and Web of Science Conference Proceedings Citation index, among others.

Awards

Ada-Europe will offer honorary awards for the best regular paper and the best presentation.

Call for Industrial Presentations

The conference seeks industrial presentations which deliver value and insight but may not fit the selection process for regular papers. Authors are invited to submit a presentation outline of exactly 1 page in length by January 25, 2015. Submissions shall be made via EasyChair following the link https://easychair.org/conferences/?conf=adaeurope2015. The format for submission is solely PDF.

The Industrial Committee will review the submissions and make the selection. The authors of selected presentations shall prepare a final short abstract and submit it by April 12, 2015, aiming at a 20-minute talk. The authors of accepted presentations will be invited to submit corresponding articles for publication in the Ada User Journal (http://www.ada-europe.org/auj/), which will host the proceedings of the Industrial Program of the Conference. For any further information please contact the Industrial co-Chairs directly.

Call for Tutorials

Tutorials should address subjects that fall within the scope of the conference and may be proposed as either half- or full-day events. Proposals should include a title, an abstract, a description of the topic, a detailed outline of the presentation, a description of the presenter's lecturing expertise in general and with the proposed topic in particular, the proposed duration (half day or full day), the intended level of the tutorial (introductory, intermediate, or advanced), the recommended audience experience and background, and a statement of the reasons for attending. Proposals should be submitted by e-mail to the Tutorial Chair.

The authors of accepted full-day tutorials will receive a complimentary conference registration as well as a fee for every paying participant in excess of 5; for half-day tutorials, these benefits will be accordingly 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 week. Workshop proposals should be submitted to the Conference Chair. The workshop organizer shall also commit to preparing proceedings for timely publication in the Ada User Journal.

Call for Exhibitors

The commercial exhibition will span the three 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.

Grants for Reduced Student Fees

A limited number of sponsored grants for reduced fees is expected to be available for students who would like to attend the conference or tutorials. Contact the Conference Chair for details.

Organizing Committee

Conference Chair

Alejandro Alonso, Universidad Politécnica de Madrid, Spain, aalonso@dit.upm.es

Program co-Chairs

Juan A. de la Puente, Universidad Politécnica de Madrid, Spain, jpuente@dit.upm.es

Tullio Vardanega, Universita di Padova, Italy, tullio.vardanega@unipd.it

Tutorial Chair

Jorge Real, Universitat Politècnica de València, Spain, jorge@disca.upv.es

Exhibition Chair

Santiago Urueña, GMV, Spain, suruena@gmv.com

Industrial co-Chairs

Jorgen Bundgaard, Ramboll, Denmark, jogb@ramboll.dk

Ana Rodríguez, Silver Atena, Spain, ana.rodriguez@silver-atena.es

Publicity Chair

Dirk Craeynest, Ada-Belgium & KU Leuven, Dirk.Craeynest@cs.kuleuven.be

Local Chair

Juan Zamorano, Universidad Politécnica de Madrid, Spain, jzamora@fi.upm.es

Program Committee

- Mario Aldea, Universidad de Cantabria, Spain
- Ted Baker, NSF, USA
- Johann Blieberger, Technische Universität Wien, Austria
- Bernd Burgstaller, Yonsei University, Korea
- Alan Burns, University of York, UK
- Maryline Chetto, University of Nantes, France
- Juan A. de la Puente, Universidad Politécnica de Madrid, Spain
- Laurent George, ECE Paris, France
- Michael González Harbour, Universidad de Cantabria, Spain
- J. Javier Gutiérrez, Universidad de Cantabria, Spain
- Jérôme Hugues, ISAE, France
- Hubert Keller, Institut für Angewandte Informatik, Germany
- Albert Llemosí, Universitat de les Illes Balears, Spain
- Franco Mazzanti, ISTI-CNR, Italy
- Stephen Michell, Maurya Software, Canada
- Jürgen Mottok, Regensburg University of Applied Sciences, Germany
- Laurent Pautet, Telecom ParisTech, France
- Luís Miguel Pinho, CISTER/ISEP, Portugal
- Erhard Plödereder, Universität Stuttgart, Germany
- Jorge Real, Universitat Politècnica de València, Spain
- José Ruiz, AdaCore, France
- Sergio Sáez, Universitat Politècnica de Valencia, Spain
- Amund Skavhaug, NTNU, Norway
- Tucker Taft, AdaCore, USA
- Theodor Tempelmeier, University of Applied Sciences Rosenheim, Germany
- Elena Troubitsyna, Åbo Akademi University, Finland
- Santiago Urueña, GMV, Spain
- Tullio Vardanega, Università di Padova, Italy

Industrial Committee

- Roger Brandt, Roger Brandt IT Konsult AB, Sweden
- Ian Broster, Rapita Systems, UK
- Jørgen Bundgaard, Rambøll Denmark A/S

- Dirk Craeynest, Ada-Belgium & KU Leuven, Belgium
- Peter Dencker, ETAS GmbH (retired), Germany
- Ismael Lafoz, Airbus Defence and Space, Spain
- Ahlan Marriott, White Elephant, Switzerland
- Steen Palm, Terma, Denmark
- Paolo Panaroni, Intecs, Italy
- Paul Parkinson, Wind River, UK
- Eric Perlade, AdaCore, France
- Martyn Pike, Embedded Consulting UK Ltd, UK
- Ana Rodríguez, Silver-Atena, Spain
- Jean-Pierre Rosen, Adalog, France
- Florian Schanda, Altran UK, UK
- Jacob Sparre Andersen, JSA Consulting, Denmark
- Claus Stellwag, Elektrobit AG, Germany
- Jean-Loup Terraillon, European Space Agency, the Netherlands
- Rod White, MBDA, UK

FOSDEM 2015

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Sat, 29 Nov 2014 20:42:19 +0000
Subject: Ada Developer Room at FOSDEM 2015 - deadline Sun 14 Dec 2014
Newsgroups: comp.lang.ada, fr.comp.lang.ada

Call for Presentations

6th Ada Developer Room at FOSDEM 2015

Saturday 31 January 2015, Brussels, Belgium

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

Organised in cooperation with Ada-Europe

Deadline Sunday 14 December 2014

The 6th Ada Developer Room will take place on Saturday 31 January at FOSDEM 2015 in Brussels, Belgium.

The Call for Presentations is still open: the extended deadline is Sunday 14 December 2014.

Do you have a talk you want to give?

Do you have a project you would like to present?

Would you like to get more people involved with your project?

We're inviting proposals that are related to Ada software development, and include a technical oriented discussion. You're not limited to slide presentations, of course. Be creative. Propose something fun to share with people so they might feel some of your enthusiasm for Ada!

Speaking slots are 25 or 50 minutes, including Q&A. Depending on interest, we might also organise a session with lightning presentations (e.g. 5 minutes each).

Please provide a proposed title, the preferred length, plus an abstract and a short bio similar in style as on the program for previous Ada DevRooms, see for example [1]. We need that to put the draft program together mid December.

Please react ASAP, and submit proposals by December 14, 2014 at the latest.

We're looking forward to more proposals!

Dirk Craeynest, FOSDEM Team of Ada-Belgium

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

PS: The full Call for Presentations is posted on the dedicated web-page on the Ada-Belgium site [2].

[1] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/14/140201-fosdem.html

[2] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Wed, 14 Jan 2015 21:36:30 +0000
Subject: FOSDEM 2015 - Ada Developer Room - Sat 31 Jan 2015 - Brussels
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

Ada-Belgium is pleased to announce its

Ada Developer Room at FOSDEM 2015

(Ada at the Free and Open source Software Developers' European Meeting)

Saturday 31 January 2015

Université Libre de Bruxelles (U.L.B.), Solbosch Campus, Room S.AW1.124

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

Organised in cooperation with Ada-Europe

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

http://fosdem.org/2015/schedule/track/ada

The Free and Open source Software Developers' European Meeting (FOSDEM) is an annual event held in Brussels, Belgium, around early February. The 2015 edition takes place on Saturday the 31st of January and Sunday the 1st of February. Ada-Belgium has organised a series of presentations related to Ada, to be held in a dedicated Developer Room, on the first day of the event.

[...]

This DevRoom aims to present the capabilities offered by the Ada language (object-oriented, multicore, embedded programming) as well as some of the many exciting tools and projects using Ada.

Ada Developer Room Presentations (S.AW1.124, 59 seats)

The Ada DevRoom program starts after the opening FOSDEM keynote, runs from 11:00 to 19:00, and consists of 7 hours with 9 talks/demos by 8 presenters from 5 different countries, plus 2 half-hour breaks with informal discussions.

10:30-11:00 - 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.

11:00-11:05 - Welcome

by Dirk Craeynest - Ada-Belgium

Welcome to the Ada Developer Room at FOSDEM 2015, which is organised by Ada-Belgium in cooperation with Ada-Europe. Ada-Belgium and Ada-Europe are non-profit organisations 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-organisations, such as Ada-Belgium, in various countries. More information on this DevRoom is available on the Ada-Belgium web-site (see URL above).

11:05-11:55 - Ada, an Introduction

by Jérémy Rosen - Open Wide

This talk will introduce the Ada programming language to people used to more classical, weak-typed languages. We will focus on how Ada uses its strong typing basis to prevent the most common programming errors at the language level, allowing the compiler to check them before they cause problems.

12:00-12:50 - Building a GUI for an Ada Application with GtkAda

by Serge Vanschoenwinkel - Eurocontrol

GTK+ is an open-source library that allows to quickly and easily build a graphical user interface, using standard widgets like buttons, combo boxes, text and tree views, scroll bars, etc. Even though GTK+ is written in C, it can be used from an Ada application thanks to GtkAda, an object-oriented Ada/C binding. Illustrated by a poker game application, this presentation will explain the essential concepts of GtkAda. It will show how to create the most common widgets and how to interact with the user.

13:00-13:25 - Opening the Development of PHCpack

by Jan Verschelde - University of Illinois at Chicago

PHCpack originated from bundling programs to solve polynomial systems with symbolic-numeric and polyhedral methods. The core of PHCpack consists mainly of Ada code, with interfaces to C and Python. Its blackbox solver is accessible from various scientific software packages such as Macaulay2, Maple, MATLAB, Octave, and Sage. The goal of the talk is to explain the application of software engineering principles and the role of Ada in the development of PHCpack.

13:30-14:00 - Informal Discussions

A half-hour slot has been reserved for much needed interaction and informal discussion among Ada DevRoom participants and anyone potentially interested in Ada.

14:00-14:50 - Contract-based Programming - A Route to Finding Bugs Earlier

by Jacob Sparre Andersen - JSA Research & Innovation

Contract-based programming is a software development technique, which is used to find programming errors earlier in the development process. "Contract" refers to formal declarations of how types and subprograms ("functions and methods" if you aren't an Ada programmer already) behave. In the strictest form, the contracts are checked as a part of the compilation process, and only a program which can be proven to conform with the contracts will compile. In a less strict form, it is more similar to "preventive debugging", where the contracts are inserted as run-time checks, which makes it more likely to identify errors during testing. Ada provides a quite extensive support for contract-based programming. The checks are specified as a mix of compile-time checks, obligatory run-time checks, and optional run-time checks. In addition to that, SPARK defines a subset of Ada with full compile-time checks.

The presentation will introduce the Ada features related to contract-based programming, and provide suggestions for how to make use of the features in practice. It is organised in three main sections: type/object invariants; pre- and postconditions for operations; making the contracts for entire packages consistent. If there is time, the presentation will close with a live test of the guidelines on an example problem selected by the audience. The intended audience is anybody with enough programming experience to know concepts like types, encapsulation and packages. Having seen source text in Pascal-like programming languages will be a benefit.

15:00-15:50 - Ada for ARM Bare Board

by Tristan Gingold - AdaCore

In 2014, AdaCore has released two new components in the GNAT GPL Edition: GNAT GPL for ARM Bare Board and SPARK 2014. I present the content of GNAT GPL for ARM, its Ravenscar runtime, how to build and deploy an embedded application in Ada and how it was used to teach Ada. Two different demos will be presented: a Tetris game and a train signalling system. Both are fully written in Ada, with some parts written and proven with SPARK 2014.

16:00-16:50 - Multithreading Made Easy, part 3

- Bounded Work Queues

by Ludovic Brenta - Debian Project

Ada is one of very few programming languages that support multithreading as part of the language, as opposed to libraries. In the previous two episodes, we showed how Ada makes it easy to turn a single-threaded program into a multi-threaded program. We ended up with ten thousand threads working concurrently then introduced a task pool and work queue wherein a small number of threads (one per processor core) process thousands of small work units. But the work queue could become very big. In this third and last episode, we show how to restrict the size of the work queue to a fixed limit, thereby preventing denial-of-service attacks.

This presentation will feature live editing of source code, compilation and debugging. Questions from beginners are encouraged. It is not necessary to have attended the first installments. The sources of our example program will be provided to those who want to tinker with them.

17:00-17:50 - 2D Drawing with Ada and Cairo

by Serge Vanschoenwinkel - Eurocontrol

Cairo is a 2D graphics library with support for multiple output devices. It is designed to produce consistent output on all output media while taking advantage of display hardware acceleration when available. The Cairo API provides operations similar to the drawing operators of PostScript and PDF. Operations in Cairo including stroking and filling cubic Bézier splines, transforming and compositing translucent images, and antialiased text rendering. All drawing operations can be transformed by any affine transformation (scale, rotation, shear, etc.). Illustrated by a poker game application, this presentation will show you how to do nice drawings with Cairo, still programming with your preferred language: Ada!

18:00-18:25 - Building Economic Simulations in Ada

by Graham Stark - Virtual Worlds Research

Virtual Worlds Research has been using Ada to build large scale economic simulations for 10 years now. These simulations have been used by Governments and others to model the effects of, amongst other things, changing Legal Aid and reforming Social Care funding - many billions of pounds of annual spending. Here, I discuss our experiences, good and bad, with the Ada language, and provide a live demonstration of the most recent model. I'll also discuss work in progress to build a new forecasting model in association with the University of Southampton.

18:30-19:00 - Informal Discussions & Closing

Informal discussion on ideas and proposals for future events.

More information on Ada DevRoom

Speakers bios, pointers to relevant information, links to the FOSDEM site, etc., are available on the Ada-Belgium site at http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

We invite you to attend some or all of the presentations: they will be given in English. Everybody interested can attend FOSDEM 2015; no registration is necessary.

We hope to see many of you there!

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Tue, 3 Feb 2015 21:55:51 +0000
Subject: FOSDEM 2015 - Presentations Ada Developer Room on-line
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

** All presentations available on-line **

Ada Developer Room at FOSDEM 2015

(Ada at the Free and Open source Software Developers' European Meeting)

Saturday 31 January 2015

Université Libre de Bruxelles (U.L.B.), Solbosch Campus, Room S.AW1.124

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

Organised in cooperation with Ada-Europe

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

---

All presentations from our 6th Ada Developer Room, held at FOSDEM 2015

in Brussels recently, are available on the Ada-Belgium web site now.

- "Welcome"

by Dirk Craeynest - Ada-Belgium

- "Ada, an Introduction"

by Jérémy Rosen - Open Wide

- "Building a GUI for an Ada Application with GtkAda"

by Serge Vanschoenwinkel - Eurocontrol

- "Opening the Development of PHCpack"

by Jan Verschelde - University of Illinois at Chicago

- "Contract-based Programming - A Route to Finding Bugs Earlier"

by Jacob Sparre Andersen - JSA Research & Innovation

- "Ada for ARM Bare Board"

by Tristan Gingold - AdaCore

- "Multithreading Made Easy, part 3 - Bounded Work Queues"

by Ludovic Brenta - Debian Project

- "2D Drawing with Ada and Cairo"

by Serge Vanschoenwinkel - Eurocontrol

- "Building Economic Simulations in Ada"

by Graham Stark - Virtual Worlds Research

Presentation abstracts, copies of slides, speakers bios, pointers to relevant information, links to other sites, etc., are all available on the Ada-Belgium site at:

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

Shortly, some pictures and video recordings will be posted as well. If you have additional 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 the many participants for their interest, and thanks to everyone for another nice experience!

[See also “FOSDEM 2015”, AUJ 35-4, p. 212. —sparre]

Happy Birthday

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Wed, 10 Dec 2014 06:48:34 +0000
Subject: Happy Birthday Ada Lovelace
Newsgroups: comp.lang.ada, fr.comp.lang.ada

Happy Birthday Ada Lovelace

Today, December 10 2014, is the 199th birthday of Augusta Ada Byron, aka Lady Ada Lovelace, recognised by many as the first programmer.

The programming language Ada was named in her honour.

I'd like to remind everyone that a promotional image combining the historical figure of Ada Lovelace and the programming language Ada is available at [1].

Useful to know is that an "Ada Mascot Competition" [2] is ongoing. It would be particularly fitting if some of the proposals would be based on a modernised Ada Lovelace image. Not many programming languages have the name of the first female programmer... ;-)

Finally, remember that deadlines for submissions to several Ada events are upcoming: the Ada Developers Room [3] at FOSDEM 2015 in Brussels, and the Ada-Europe 2015 conference [4] in Madrid, Spain. Share your experience at an Ada event in 2015, the 200th anniversary year of Ada Lovelace!

Happy birthday, Lady Ada!

Dirk Craeynest

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

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

[2] http://www.gnoga.com/rebirth.html

[3] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html

[4] http://www.ada-europe.org/conference2015

Mascot Competition

From: David Botton <david@botton.com>
Date: Thu, 29 Jan 2015 19:57:50 -0800
Subject: Ada Mascot Contest - Deadline moved to March 13
Newsgroups: comp.lang.ada

The Ada Mascot competition deadline has been moved to March 13 a month later to accommodate FOSDEM entries.

http://gnoga.com/mascot.html

Let your artistic friends know! Prize is at $600 currently.

[Later updated to 700 USD. —sparre]

[See also “Ada-rebirth — The Ada Mascot Competition”, AUJ 35-4, p. 233. —sparre]

Ada Semantic Interface Specification (ASIS)

The State of ASIS

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 2 Jul 2014 17:06:28 -0500
Subject: Re: a new language, designed for safety !
Newsgroups: comp.lang.ada

> [...]

Since the vendors were ignoring what the ARG was trying to do, and there is very little use of "standard" ASIS anyway (most applications only work with one implementation -- we really only need a standard for applications that will be ported from one implementation to another), it isn't worth anyone's effort to make a standard. Vendors claim that their customers don't care about a standard for ASIS. If ASIS customers were to demand that their vendors supported a standard, then things would be different, but there is no evidence of that (beyond Mr. Rosen -- which is surely not enough).

The ASIS 95 standard is quite a mess; it wouldn't be possible to create an ASIS implementation just from reading the standard. One would have to see what other implementations do. (That's true to some extent for all standards, but the ASIS standard is worse. And, at least for the Ada standard we have the ACATS to provide some additional insight into what a correct implementation needs to do.)

Ada-related Resources

Inspiration for the Mascot Competition

From: Vincent Diemunsch <vincent.diemunsch@gmail.com>
Date: Sat, 15 Nov 2014 01:22:54 -0800
Subject: Re: The Owl - My Idea for the Ada mascot
Newsgroups: comp.lang.ada

[...]

For an Ada mascot, I like the idea of a bird, for the same kind of reasons you gave. But as others said, the owl sounds quite a nasty and disturbing bird, even if the ancient Greeks took them for a symbol of wisdom : "their eyes are directed forward like those of human beings" and "Most of them utter a hooting cry like a groan, and as they inhabit ruins, they sound as though mourning over the devastation, and hence symbolise in the Bible destruction and desolation"...

Therefore, even if the owl is tempting, a mascot related to "Destruction and Desolation" is not possible for Ada which is already associated with DoD and fighter...

I don't really like the Aardvark, sort of pig which lives from bugs. Sounds geek, clumsy, harmless but dirty... Definitely Not Ada (or maybe for Ada 95 perhaps.... :-)).

My proposal : a small, nice, gentle, little bird called ... Ada in French !!!

http://fr.wikipedia.org/wiki/Ada_(oiseau)

It is light, swift, nice, harmless, precise, poetic.

- Light: because Ada, as Ichbiah said, must stay a simple language, with fast and inexpensive compilers, enabling cost effective developments.
- Swift: because it must have good runtime performances.
- Nice: necessary for a Mascot!
- Harmless: not related to war.
- Precise: it's a bird, not a pig.
- Poetic: Ada was the daughter of Lord Byron, a poet. A small bird is a poetic character: it relates to freedom and art.

It is also GREEN (in the sense of environment friendly) for it is a real bird!

And finally the image of a bird for a programming language is fashionable: Apple has taken a bird for the logo of its new language: Swift: https://developer.apple.com/swift/

Now I let someone gifted in drawing, create a logo and submit it for the contest.

From: Vincent Diemunsch <vincent.diemunsch@gmail.com>
Date: Sat, 15 Nov 2014 01:36:26 -0800
Subject: Re: The Owl - My Idea for the Ada mascot
Newsgroups: comp.lang.ada

To add to my previous message on an Ada Mascot idea, here is a view of the "Ada à bec bleu" or Blue-billed Black Tyrant in English, an American bird:

http://en.wikipedia.org/wiki/Blue-billed_black_tyrant#mediaviewer/File:Knipolegus_cyanirostris_-Reserva_Guainumbi,_Sao_Luis_do_Paraitinga,_Sao_Paulo,_Brasil-8.jpg

From: David Botton <david@botton.com>
Date: Sat, 15 Nov 2014 18:46:18 -0800
Subject: Re: The Owl - My Idea for the Ada mascot
Newsgroups: comp.lang.ada

> [...]

Hmmm, I like the idea. The Rush Tyrant - http://www.pinterest.com/pin/282741682827752439 even meets the cuddly cute idea many have for a mascot.

Even though the contest will hopefully get us a nice general mascot various plays on it for emblems, logos etc. are possible and the "Ada" (Tyrant) has a lot of variety.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Sat, 15 Nov 2014 19:19:42 +0200
Subject: Re: The Owl - My Idea for the Ada mascot
Newsgroups: comp.lang.ada

> [...]

For an Ada emblem, I suggest a twig of a wine, with some green wine leaves and a bunch of grapes.

Connections to Ada:

- Green
- Long-lived, portable, international
- Produces good, enjoyable things (= grapes and wine)
- Originates from embedded systems (= roots)
- Cluster of grapes = parallel tasks
- Developed (= cultured and selected) with care and intelligence
- Strong quality control and verification
- The Beaujolais Connection
- Runs on sunlight, as many Ada programs do.

From: Natasha Porté <lithiumcat@instinctive.eu>
Date: Sat, 22 Nov 2014 10:02:04 +0000
Subject: Re: The Owl - My Idea for the Ada mascot
Newsgroups: comp.lang.ada

> Call me a pessimist, but any animal would be a bad representation for Ada; they evolve, much like C programs, over the course of many years by hasty and unplanned enhancements, included in a reactionary and haphazard manner based solely on what caused the previous one to fail.

What about a robot then? Cold like proper engineering, unforgiving like an Ada compiler, durable like well-engineered constructs (though it could still be made cute, like Wall-E).

From: Jerry van Dijk <jdijk59@hotmail.com>
Date: Sat, 22 Nov 2014 13:36:25 +0100
Subject: Re: The Owl - My Idea for the Ada mascot
Newsgroups: comp.lang.ada

> [...]

Durable like Marvin ? :-)

Anyway I find it hard to come up with anything as the first thing that comes to my mind if you mention Ada is 'Solid'.

Maybe simply something like a big strong hammer? I'm sure someone can create a cute fluffy version too...

New Ada motto: to Ada all programming problems ARE nails :-)

Ok, ok, I'm going to get my coffee now...

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 02 Dec 2014 07:00:55 +0100
Subject: Re: The Owl - My Idea for the Ada mascot
Newsgroups: comp.lang.ada

Eryndlia Mavourneen wrote:

> ... Could have a little hammer in its hand. lol

... or rather multiple arms (like Shiva), each holding a different tool (we don't have only one tool)

From: David Botton <david@botton.com>
Date: Fri, 21 Nov 2014 10:23:25 -0800
Subject: Lady Ada Statue
Newsgroups: comp.lang.ada

I came across this looking for ideas for Mascot and another site I am working on (You will see it soon ;)

Statue of Lady Ada

http://www.tracyhsugg.com/commission/ada.php

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Fri, 21 Nov 2014 13:57:03 -0700
Subject: Re: Lady Ada Statue
Newsgroups: comp.lang.ada

> [...]

Also the SIGAda award statuette:

http://www.sigada.org/exec/awards/awards.html

From: Tom Moran <tmoran@acm.org>
Date: Sat, 22 Nov 2014 05:49:26 +0000
Subject: Ada Lovelace doll
Newsgroups: comp.lang.ada

I see Miss Possible on Indiegogo has an Ada Lovelace doll. (They focus on female role model dolls.)

GetAdaNow.com

From: David Botton <david@botton.com>
Date: Sat, 22 Nov 2014 21:30:43 -0800
Subject: GetAdaNow.com
Newsgroups: comp.lang.ada

As part of the effort to start advocating Ada to the applications space, I've set up a new website:

http://GetAdaNow.com

It is designed to make it as easy as currently possible to get new developers set up to use Ada and pointed to some updated resources.

If you have any suggestions please let me know.

Please add links from your sites to http://GetAdaNow.com to help people use your projects.

AdaPower will be redone soon and have a way for you to maintain your own listings of projects, articles, etc. and will have multiple maintainers so it can stay up to date and not depend on any one person.

Posters from SIGAda

From: Michael Feldman <mfeldman@gwu.edu>
Date: Fri, 05 Dec 2014 15:29:01 -0800
Subject: Re: Ada/SPARK in railway signalling
Newsgroups: comp.lang.ada

> [...]

Have a look at that home page now. Posters are up for Victoria Line and Paris Line 1. Also in the full gallery at:

http://sigada.org/awareness/ada-posters-gallery/index.html

Just click on a thumbprint to get a full resizable printable poster.

Getting Started with Ada

From: David Botton <david@botton.com>
Date: Mon, 8 Dec 2014 16:13:28 -0500
Subject: Documentation
Newsgroups: gmane.comp.lang.ada.gnoga

Slowly but surely working on documentation...

I've checked in my start of "Getting Started with Ada" at docs/learn_ada.md [In the Gnoga source text repository. —sparre]

It is intended to be a quick and dirty intro to Ada for someone with some programming experience. When done a programmer won't know all of Ada but should know enough to write basic Ada apps, i.e. work with Gnoga to do what most people do in VB or Delphi.

My goal is to just make Ada easy enough to take away a bit the fear factor of considering Gnoga for someone's next project instead of say Node.js, PHP or Ruby, etc.

If someone has a few minutes here or there and can comment appreciated. In particular if they think it is short enough but enough for some Web developer they may know to get enough of an idea of Ada to try it.

Analytical Engine Emulator

From: Brad Moore <brad.moore@shaw.ca>
Date: Wed, 10 Dec 2014 15:04:56 -0700
Subject: Re: Happy Birthday Ada Lovelace
Newsgroups: comp.lang.ada, fr.comp.lang.ada

[On the 199th birthday of Augusta Ada Byron.]

> Has anyone ever attempted to create a software simulation of the Analytical Engine? It would be great if we could actually execute the Lady Lovelace's programs, even if only in a virtual machine. Why do I have the feeling she would have appreciated the elegance of that?

There is an online java applet you can try here....

https://www.fourmilab.ch/babbage/applet.html

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 10 Dec 2014 14:14:00 +0100
Subject: Re: Happy Birthday Ada Lovelace
Newsgroups: comp.lang.ada, fr.comp.lang.ada

> [...]

There is an attempt to create a real Analytical Engine, see

http://plan28.org

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Wed, 10 Dec 2014 13:52:53 -0500
Subject: Re: Happy Birthday Ada Lovelace
Newsgroups: comp.lang.ada, fr.comp.lang.ada

> [...]

Cool! Now we need an Ada compiler that targets the Analytical Engine.

Ada on Social Media

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Tue Feb 24 2015
Subject: Ada on Social Media

Ada groups on various social media:

- LinkedIn[1]: 2_168 members
- Reddit[2]: 770 readers
- Google+[3]: 439 members
- StackOverflow[4]: 268 followers
- Twitter[5]: 4 tweeters

[1] http://www.linkedin.com/groups?gid=114211

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

[3] https://plus.google.com/communities/102688015980369378804

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

[5] https://twitter.com/search?f=realtime&q=%23AdaProgramming

[See also “Ada on Social Media”, AUJ 35-4, p. 215. —sparre]

Open Source Build Server Status

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue Feb 24 2015
Subject: Jenkins
URL: http://build.ada-language.com/

[Builds: —sparre]

- Ahven_JNT
- Ahven_Win7_GNAT2013
- Ahven_Win7_ICCAda
- JD_JNT
- Jdaughter - Debian 7.0 - GNAT 4.6
- Jdaughter_Win7_ICCAda
- Lace_Win7_ICCAda

[Fails to build: —sparre]

- Ahven - Debian 7.0 - GNAT 4.6
- AVR-Ada_Debian_7
- Strings_Edit_ICCAda
- UnzipAda_Win7_GNAT2013
- UnzipAda_Win7_ICCAda

[See also “Open Source Build Server Status”, AUJ 35-4, p. 215. —sparre]

Repositories of Open Source Software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Tue Feb 24 2015
Subject: Repositories of Open Source software

AdaForge: 8 repositories [1]

Bitbucket: 109 repositories [2]
16 developers [2]

BlackDuck OpenHUB: 208 projects [3]

Codelabs: 20+ repositories [4]

GitHub: 749 repositories [5]
179 developers [6]

OpenDO Forge: 24 projects [7]
424 developers [7]

Rosetta Code: 612 examples [8]
28 developers [9]

Sourceforge: 243 repositories [10]

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

[2] http://edb.jacob-sparre.dk/Ada/on_bitbucket

[3] https://www.openhub.net/tags/ada

[4] http://git.codelabs.ch/

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

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

[7] https://forge.open-do.org/

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

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

[10] http://sourceforge.net/directory/language%3Aada/

[See also “Repositories of Open Source Software”, AUJ 35-4, p. 214. —sparre]

Ada-related Tools

GNAT

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 14 May 2014 16:48:19 -0500
Subject: Re: Bug or feature?
Newsgroups: comp.lang.ada

> [...]

To get what the Ada Standard calls "standard mode" for Ada with GNAT, you need to compile with a bunch of options. The default behaviour of GNAT is NOT standard mode as described in the RM.

To compile ACATS tests in GNAT, I have to use a small boatload of options:

gnatmake C457003.adb -eS -gnat12 -O0 -gnatE -gnato -gnatv -gnatws -gnatd7 -bargs - T0

Some of these are about warnings and disabling of optimisations, and of course "-gnat12" sets Ada 2012 mode (which I think is the default these days). [B-Tests also need -gnatf and -gnatq, but that's not important to most since they're not worried about

I use that set of options anytime I'm compiling test programs from GNAT (including the ones that appear here), because several of them are needed to get standard behaviour. GNAT's default behaviour might be "better" in some ways, but it's confusing because it doesn't necessarily do what you'll find in an Ada book.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Wed, 14 May 2014 18:35:05 -0400
Subject: Re: Bug or feature?
Newsgroups: comp.lang.ada

> [...]

Of those, only "-gnatE", "-gnato", and "-bargs -T0" are needed for standards conformance. And as I said, "-gnato" is no longer, or soon will no longer be needed.

The above options are definitely not what the OP should be using. For example, don't use -gnatE unless you are porting a large program to GNAT and it won't work otherwise.

> [...]

Right, Ada 2012 is now the default.

[...]

From: Simon Wright <simon@pushface.org>
Date: Thu, 15 May 2014 09:23:04 +0100
Subject: Re: Bug or feature?
Newsgroups: comp.lang.ada

> [...]

-gnato isn't the default for GNAT GPL 2014 or FSF GCC 4.9.0.

Add -fstack-check ?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 15 May 2014 13:21:27 -0500
Subject: Re: Bug or feature?
Newsgroups: comp.lang.ada>

[...]

> Add -fstack-check ?

That's probably a good idea for real programs. The ACATS doesn't need it as the tests that used to attempt to exhaust memory have been reigned in. (They had nasty effects on targets supporting virtual memory. I remember the first time we ran the ACATS on Windows NT, one of those tests allocated an insane amount of swap space and then essentially ran from the disk drive. It would have taken months to complete. I had to artificially bound the heap size on NT in order to eliminate that problem; that's the sort of counterproductive thing that one would hope the ACATS is not requiring.) The tests now allocate a few megabytes and then give up - thus on virtual memory hosts they don't try to test for Storage_Error.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 15 May 2014 10:58:02 +0200
Subject: Re: Bug or feature?
Newsgroups: comp.lang.ada

On 14/05/14 23:48, Randy Brukardt wrote:

> To compile ACATS tests in GNAT, I have to use a small boatload of options:
>

> gnatmake

> C457003.adb -eS -gnat12 -O0 -gnatE -gnato -gnatv -gnatws -gnatd7 -bargs - T0

Would it be meaningful when testing any compiler, to include

the optimisers typically used when translating production code?

-O2 seems commonly used with GNAT.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 15 May 2014 13:30:41 -0500
Subject: Re: Bug or feature?
Newsgroups: comp.lang.ada

> [...]

I picked these options originally because they were the ones used during the latest formal conformity assessment of GNAT. (I've since modified them a bit on the advice of the AdaCore ACATS test person.) That's the only set of options that anyone ever guaranteed actually met the Standard.

One would expect that internally, AdaCore tests other sets of options as well. Optimisation is sometimes a problem, as really powerful optimisers can sometimes eliminate or invalidate ACATS tests. ACATS tests have been repaired to avoid optimisation effects, but it's a never-ending game of whack-a-mole. (As optimisers get better, new problems emerge, which require still more test repairs, etc.) In addition, some optimisation modes probably aren't standards-conformant. (For instance, Janus/Ada has a mode where all objects are assumed to be in range. This matches our Ada 83 compiler, but it's not correct for Ada 95 and later.)

For Janus/Ada, I run 3 different sets of optimisation options, combined with 3 different language settings. But that's for in-house use only; a formal conformity assessment would be done with the optimisation off. The in-house goal is to minimise failures with the optimiser on but there are a few failures that are effectively unfixable, so I doubt it would ever be perfect.

From: David Botton <david@botton.com>
Date: Wed, 12 Nov 2014 06:33:42 -0800
Subject: Re: What exactly is the licensing situation with GNAT?
Newsgroups: comp.lang.ada

[Janus/Ada and CLAW]

However, GWindows and GNATCOM (http://sourceforge.net/projects/gnavi/) are more capable, very well maintained, open source, and easier to use and they work well with the windows FSF version of GNAT that comes with MinGW. You can use Ada 2005 and Ada 2012 with them as a result too.

Despite the lack of PR (not sure why they are not listed in the AdaIC.org list of packages for example. It is probably the largest Ada framework used outside of those distributed today by AdaCore in the world.

When I created them they were placed under the GMGPL so you can enjoy using them in commercial products and I have and hundreds others do.

When I abandoned public Ada projects and advocacy ten years ago, _because_ those libraries are open source people like Gautier de Montmollin and others were willing to get involved and are even running the show on them now. So Open Source is a critical part of success today. You just have to be creative to know how to monetise it.

For application development MinGW is absolutely usable for professional development.

Same goes for other platform GNAT's and you should have no qualms of using it for non-safety critical work.

Request: Compiler for XMOS Controllers

From: Erlo
Date: Sun, 09 Nov 2014 16:26:37 +0100
Subject: Ada for XMOS?
Newsgroups: comp.lang.ada

Anybody working on a Ada-compiler for XMOS controllers? (www.xmos.com)

NC_Socket

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 14 Nov 2014 16:29:59 -0600
Subject: Re: What exactly is the licensing situation with GNAT?
Newsgroups: comp.lang.ada

[...]

I've been using Claw.Sockets for so long that I don't know what the underlying implementation is. So I don't know what "select" is used for or whether it's implemented in NC_Sockets.
Claw.Sockets has a server type that's used for implemented servers (like web and mail servers).

As to the platforms, Linux 64-bit (tested under GNAT) and Windows 32-bit (tested under GNAT and Janus/Ada) are what I have in hand. I suspect that Linux 32-bit would be easy to create based on the 64-bit version, but I've had no need to do so.

I'll be posting a NC_Sockets package fairly soon, once I get the last few spec updates finished and correct the test programs. (And I have to figure out error handling in the Linux version, I don't think it works right, and that's a big deal to my servers of course.) My best guess is that I'll get this done in January or so (have some ARG and ACATS tasks to do first).

Request: Video Decoding Library

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Sun, 16 Nov 2014 16:12:51 +0100
Subject: Video decoding library?
Newsgroups: comp.lang.ada

Are there any Ada libraries for decoding (compressed) video files? At the moment I'm not particular about which formats should be supported.

Ada bindings to libraries written in other languages are also of interest, even if I would like a pure Ada solution.

I'm aware of "ada-ffmpeg"[1], but it looks like an extremely thin binding to a C library. The Ada binding to "OpenCV"[2] seems to include a partial "ffmpeg" binding as well, but it doesn't look significantly more elegant at a first glance.

[1] https://github.com/xlq/ada-ffmpeg

[2] https://code.google.com/p/opencvada/

Gnoga

From: David Botton <david@botton.com>
Date: Sun, 16 Nov 2014 20:02:21 -0500
Subject: Source Code View for Gnoga Website
Newsgroups: gmane.comp.lang.ada.gnoga

If you go to:

http://www.gnoga.com/source

You can see the source code for snake and the website.

From: David Botton <david@botton.com>
Date: Tue, 25 Nov 2014 23:12:51 -0500
Subject: Working alternative to AWS
Newsgroups: gmane.comp.lang.ada.gnoga

I have a working version now of Gnoga using Dmitry Kazakov's simple components instead of AWS. I still have a bit of polish to do, but it looks like a winner.

This means AWS not required at all (nor any of its many dependencies), a simple checkout of Gnoga will include all needed and should compile on even older versions of GNAT (although still needs a few 2012 features currently). In theory I could probably backport Gnoga to 2005 or even Ada 95 at this point.

My goal is for Gnoga to work with the current release version of MinGW on Windows and Debian Squeeze. So if I manage that I'll be super happy.

I'll post when I've got everything tested and have pushed this in to Git. It may have to wait until tomorrow though.

[See also "Simple Components", AUJ 35-4, p. 217. —sparre]

From: David Botton <david@botton.com>
Date: Thu, 27 Nov 2014 10:18:11 -0500
Subject: In Git - New Gnoga using Simple_Components HTTP and Websockets
Newsgroups: gmane.comp.lang.ada.gnoga

I have committed the work I've done to move Gnoga to Dmitry's Simple_Components. The Gnoga website and snake example are also now using it.

There have been numerous other improvements to the communication infrastructure as part of this as well.

In some ways the entire system is far more robust than before and continuing to harden on the other hand this is a brand new implementation of WebSockets and we are the first ones using it. (I am much impressed with Dmitry's creating the entire WebSockets code from specs and almost no testing, etc.)

Given that and that this is a lower level API than AWS there are occasionally some issues that come up and am working through them. Please report and issues you have. If possible also copy me on what it says in the JavaScript console of the browser.

The only thing "missing" functionality at the moment is Form PUT support (GET works) and File Upload. I will get those both working soon as well.

From: David Botton <david@botton.com>
Date: Mon, 29 Dec 2014 19:03:50 -0800
Subject: ANN: Gnoga v1.0 - The GNU Omnificent GUI for Ada
Newsgroups: comp.lang.ada

Introduction

http://gnoga.com

Gnoga is an Open Source development platform for the creation of mission critical and enterprise applications that can be deployed to the cloud, desktop or mobile devices.

Gnoga applications are written using the Open Source Gnoga framework licensed under the GPLv3 with Runtime Exceptions for creating free or proprietary software and Ada 2012, the time-tested, safe and secure programming language used for long-lived critical application development.

This releases contains the Gnoga 1.0 Framework, future releases will include additional platform tools including a full IDE and visual development environment.

Ada compilers are available for most platforms - see http://GetAdaNow.com

For more information: http://gnoga.com

Download Gnoga 1.0 at http://gnoga.com/gnoga.1.0.zip or clone the latest at:

git clone git://git.code.sf.net/p/gnoga/code gnoga

The Gnoga user guide is available at: http://www.gnoga.com/user_guide.html

Join the Gnoga E-Mail Support List: https://lists.sourceforge.net/lists/listinfo/gnoga-list

Gnoga Features

- Real-time live server push web-app technology for the web
- Native Gtk Front end for the desktop
- Native Mac OS X desktop applications that can be submitted to the App Store
- Write complex web-apps or desktop apps with no HTML or JS
- The same code base can deploy as a web-app, desktop or mobile app
- Server side and client side development is in same code base and all in Ada
- Gnoga applications are clear and easy to read and write
- Extensive concurrency support
- Integrates easily with C/C++, Python or any other server side language or library
- Bind any JavaScript based client libraries to take advantage of existing UI developments

Gnoga Platforms

- GNU/Linux
- Apple Macintosh OS X
- Microsoft Windows
- Raspberry Pi
- And any other GCC/Ada platform supporting GNAT.Sockets

Gnoga Framework Overview

1. The communication platform between the Ada code and the browser / native

- Gnoga.Server.Connection

2. Binding to the HTML5 DOM and Browser

- Gnoga.Gui.Base (Not per se a binding of Node but takes its place)
- Gnoga.Gui.Element, Gnoga.Gui.Element.* (HTML Elements)
- Gnoga.Gui.Element.Canvas - HTML 5 Canvas bindings
- Gnoga.Gui.Element.SVG - HTML SVG vector graphics

- Gnoga.Gui.Element.Multimedia - HTML 5 Audio and Video

- Gnoga.Gui.Element.Style - CSS Style blocks

- Gnoga.Gui.Window, Gnoga.Gui.Navigator, Gnoga.Gui.Screen,

- Gnoga.Gui.Location

- Gnoga.Gui.Document

3. Application start up services

- Gnoga.Server.Application.Singleton - Desktop apps

- Gnoga.Server.Application. Multi_Connect - Multi user / Web apps

4. Gnoga higher level containers and GUI widgets

- Gnoga.Gui.Views.* - Auto layout of child elements and basis for custom Gnoga Ada only widgets

- Gnoga.Gui.Views.Docker - Dock child views to view sides

- Gnoga.Gui.Views.Card - Stacks of views

5. Gnoga client side application APIs

- Gnoga.Client.Storage - local persistent and session storage on browser

- Gnoga.Client.Bind_Page - Bind to all elements on pre-made HTML5 pages

6. Gnoga database bindings and server side APIs

- Gnoga.Server.Database - support for MySQL and SQLite 3 (for ODBC bindings see deps/simple_components)

- Gnoga.Server.Model - Active Data models like in Rails

- Gnoga.Server.Migrations - Rails like database schema migrations

- Gnoga.Server.Template_Parser - Parse files with tokens or Python 2.7

7. Gnoga development tools

- tool/gnoga_make - Generate application scaffolds

8. Plugin bindings to existing JavaScript libraries

- Gnoga.Gui.Plugin.Ace_Editor - Full editor with Ada syntax highlighting

- Gnoga.Gui.Plugin.Bootstrap - The Bootstrap framework

- Gnoga.Gui.Plugin.jQuery - jQuery support to access non-Gnoga Elements

- Gnoga.Gui.Plugin.jQueryUI - all the jQueryUI Interactions and Effects

- Gnoga.Gui.Plugin.jQueryUI.Widgets - the jQueryUI Widgets

9. Native Desktop and Mobile Application Support coming:

- Gnoga.Server.Application. Gtk_Window - Native GTK front end

- Gnoga.Gui.Plugin.MacGap - Native Mac OSX features

From: David Botton <david@botton.com>
Date: Tue, 13 Jan 2015 20:31:47 -0500
Subject: Full SSL support now implemented and ready to run
Newsgroups: gmane.comp.lang.ada.gnoga

It couldn't be easier to add full direct SSL support. I have setup Jeff's Chattanooga demo in git to listen on both secure and insecure ports. I have also removed the need for a special secure boot file and now boot.js detects if connection is http or https and automatically switches the websocket protocol.

I added the following line to the gpr file:

with "../../ssl/gnoga_secure.gpr";

Then I added the following to chattanooga-ui.adb:

```
Gnoga.Server.Connection.Secure.
Register_Secure_Server(
  Certificate_File =>"/home/dbotton/
      workspace/ssl/star_gnoga_com.crt",
  Key_File     => "/home/dbotton/
      workspace/ssl/star_gnoga_com.key",
  Port       => 8443,
  Disable_Insecure => False);
Gnoga.Application.Multi_Connect.
      Initialize (Port => 8082);
```

That's it :)

This will allow you to reach the demo at http://chat.gnoga.com:8082 or at https://chat.gnoga.com:8443

I also setup on the server (as before) an ssl proxy to the non-SSL port 8082 so that you can access the demo chat at https://chat.gnoga.com

The Apache config looks like this:

<VirtualHost *:443>

ServerName chat.gnoga.com

ServerAdmin david-daM41vM3II/QT0dZR+AlfA@public.gmane.org

SSLEngine on

SSLCertificateFile /home/dbotton/workspace/ssl/star_gnoga_com.crt

SSLCertificateKeyFile /home/dbotton/workspace/ssl/star_gnoga_com.key

ProxyPass /gnoga ws://127.0.0.1:8082/gnoga

ProxyPass / http://127.0.0.1:8082/

ProxyPassReverse / http://127.0.0.1:8082/

ErrorLog ${APACHE_LOG_DIR}/gnoga.err.log

CustomLog ${APACHE_LOG_DIR}/gnoga.log common

</VirtualHost>

This gives you a complete example of SSL with Gnoga, both direct and proxy methods.

See the FAQ for how to create fake ssl certs and some tips if purchasing an SSL certificate and how to add the intermediate certificates.

From: David Botton <david@botton.com>
Date: Wed, 4 Feb 2015 21:25:30 -0500
Subject: New Feature :) - Gnoga via http long polling and AJAX - no WebSockets
Newsgroups: gmane.comp.lang.ada.gnoga

It is now possible to use http "long polling" and AJAX as an alternative to WebSockets.

Mine Detector using Ajax:

http://gnoga.com:8081/ajax.html

Snake using Ajax (works but not smooth enough to play):

http://gnoga.com:8080/ajax.html

Chattanooga using Ajax over https:

https://chat.gnoga.com/ajax.html

(The gnoga.com site needs some more changes first, but coming due to the way I use the '#' tags)

To make AJAX the default method use:

Add to your initialise the parameter boot => "ajax.html" to do so.

Note you will need to copy ajax.js to your js directory and ajax.html to your html directory of your project.

All modern browsers support WebSockets, however older corporate firewalls do not.

I do not recommend using this method for most application development as it is slightly less responsive and leaves your browser's loading icon spinning which may annoy some users. It is intended as a fallback method or to use for website development using a method somewhat like the AdaBlog demo where you use a more traditional web paradigm of stateless connectivity using local storage or other techniques to maintain state between pages. I will document more of that as we get closer to 1.1 in the coming weeks.

With this addition and a few other small additions, this will make Gnoga not only the best UI for most modern Ada applications, it also makes it the best choice for developing websites using Ada.

[See also “Gnoga”, AUJ 35-4, p. 218. —sparre]

AdaControl

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 19 Nov 2014 11:22:59 +0100
Subject: Re: What is your opinion on Global Objects?
Newsgroups: comp.lang.ada

[global variables]

Note that AdaControl (rule Directly_Accessed_Globals) can enforce

that this pattern is used safely. From the User's Guide:

"this rule enforces that all global variables are accessed by dedicated access subprograms, and that only those subprograms access the variables directly. If given with the keyword “protected” and/or “accept”, it enforces that global variables are accessed only by dedicated protected subprograms or tasks, ensuring that no race condition is possible"

Strings Edit

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 22 Nov 2014 17:16:44 +0100
Subject: ANN: Strings edit for Ada 3.1 released
Newsgroups: comp.lang.ada

The software provides I/O facilities. The following I/O items are supported by the package:

- Generic axis scales support;
- Integer numbers (generic, package Integer_Edit);
- Integer sub- and superscript numbers;
- Floating-point numbers (generic, package Float_Edit);
- Roman numbers (the type Roman);
- Strings;
- Ada-style quoted strings;
- UTF-8 encoded strings;
- Unicode maps and sets;
- Wildcard pattern matching.

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

Changes to the previous version:

- Added packages for portable stream encoding of signed and modular integer types;
- Base64 string encoding supported.

[See also “Strings_Edit”, AUJ 35-3, p. 154. —sparre]

Simple Components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 23 Nov 2014 11:20:01 +0100
Subject: ANN: Simple components v.4.3 released
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 analysers, lock-free data structures, synchronisation 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, multiple connections server designing tools.

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

Changes to the previous version:

- Persistent B-tree with keys and values allocated externally
- Persistent tables searchable by multiple keys with multiple values (columns). This can be used to design a light-weight 100% Ada equivalent of a relational DB table with several keys.
- WebSockets integrated into the HTTP server. Both half-duplex and full-duplex operating modes are supported.

From: David Botton <david@botton.com>
Date: Thu, 27 Nov 2014 07:25:47 -0800
Subject: Re: ANN: Simple components v.4.3 released
Newsgroups: comp.lang.ada

> - WebSockets integrated into the HTTP server

I am happy to announce that the latest version of Gnoga is now using Dmitry's HTTP and WebSockets components instead of AWS. In addition to being a very complete HTTP and WebSockets implementation it is lightweight and since Ada 95 it should allow Gnoga to compile on much older versions of Gnat as found currently in most distributions.

The Gnoga website http://www.gnoga.com and the snake example http://snake.gnoga.com are both running on Gnoga using the new components.

Thanks!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 20 Dec 2014 13:05:05 +0100
Subject: ANN: Simple components for Ada v4.4. released
Newsgroups: comp.lang.ada

[...]

Changes to the previous version:

- URI scheme recognition added;
- Get_Server_Address added to the connections server to allow limiting the addresses being listened;
- Socket send events are blocked when the server has no data to sent;
- Get_IO_Timeout is added to control waiting for socket events;
- Get_Polling_Timeout is added to control maximal time socket send polling remain stopped;
- Trace_Sending is added to trace socket polling events;
- Documentation improved;
- Bug fixes.

P.S. Thanks to David Botton for testing, submitting bug reports and features requests. Gnoga's use case was a quite hard test.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 17 Jan 2015 21:04:43 +0100
Subject: ANN: Simple Components for Ada v4.5 released
Newsgroups: comp.lang.ada

[...]

The new version provides bindings to GNUTLS and an implementation of SSL/TLS servers, HTTP included.

Changes to the previous version:

- Dynamically allocated terminated strings added in the package GNAT.Sockets.Connection_State_Machine.Terminated_Strings;
- Input buffer size discriminant added to HTTP_Client connection object;
- Tracing primitive operation extended for encoded/ciphered content;
- GNUTLS bindings added;
- Secure SSL/TLS multiple-connections servers added. The implementation is based on GNUTLS;
- Secure HTTP implementation added.

[See also “Simple Components”, AUJ 35-4, p. 217. —sparre]

AVR-Ada

From: Rolf Ebert <rolf.ebert.gcc@gmx.de>
Date: Mon, 24 Nov 2014 17:43:31 +0100
Subject: patches for gcc 4.9.2 updated
To: AVR-Ada <avr-ada-devel@lists.sourceforge.net>

Yesterday I pushed my patches for building the AVR-Ada cross compiler to the SF git repository. The patches are relative to gcc-4.9.2. Most are pure copies from older versions.

The patch 72-gcc-4.9-ada-timebase is necessary for correct translation of time literals in delay statements to the actual time base in AVR.Real_Time.Clock and AVR.Real_Time.Delays. I forgot that one in the previous patch sets.

From: Rolf Ebert <rolf.ebert.gcc@gmx.de>
Date: Tue, 25 Nov 2014 21:51:17 +0100
Subject: Default_Bit_Order in AVR-Ada
To: AVR-Ada <avr-ada-devel@lists.sourceforge.net>

I'm really not sure what to do here.

We have a bug report [1] that the Default_Bit_Order changed in AVR-Ada. That is true. Previously we had the Default_Bit_Order = Low_Order_First, we now have High_Order_First. As far as I remember that was for conformance to the AdaCore's AVR compiler. I actually never thought about it.

If I read the wikibook entry [2], the bug report seems correct. But I doubt that AdaCore might be incorrect in their system.ads.

Suggestions welcome

[1] http://sourceforge.net/p/avr-ada/bugs/32/

[2] http://en.wikibooks.org/wiki/Ada_Programming/Attributes/%27Bit_Order

[See also “AVR-Ada”, AUJ 35-4, p. 220. —sparre]

Open Ravenscar Kernel

From: Vincent Diemunsch <vincent.diemunsch@gmail.com>
Date: Wed, 26 Nov 2014 14:06:54 -0800
Subject: Open Ravenscar Kernel
Newsgroups: comp.lang.ada

I can't find the source code of the Open Ravenscar Kernel. The OpenRavenscar.org website is closed and I get redirected to : http://www.dit.upm.es/~ork/index.html/. Then it downloaded the GNATforLeon source package, but I couldn't find the packages implementing the ORK in Ada.

Does anybody knows where I could find them?

From: Brian Drummond <brian@shapes.demon.co.uk>
Date: Thu, 27 Nov 2014 11:34:44 +0000
Subject: Re: Open Ravenscar Kernel
Newsgroups: comp.lang.ada

> [...]

I think you'll find they are in the gcc/ada folder, not clearly separated out from the rest of the Ada compiler sources. Takes some digging to separate them out into a separate project.

Emacs Ada Mode

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Tue, 02 Dec 2014 07:09:49 -0600
Subject: ada-mode 5.1.7 now in Gnu ELPA
Newsgroups: comp.lang.ada

ada-mode 5.1.7 is now available in Gnu ELPA; mostly bug fixes. See http://stephe-leake.org/emacs/ada-mode/emacs-ada-mode.html for more info.

[See also “Emacs Ada Mode”, AUJ 35-2, p. 75. —sparre]

STM32F4 GNAT Run Time Systems

From: Simon Wright <simon@pushface.org>
Date: Sun, 07 Dec 2014 15:48:35 +0000
Subject: ANN: STM32F4 GNAT Run Time Systems 20141207
Newsgroups: comp.lang.ada

I've been working on a GNAT RTS with the GCC Runtime Library exception for STM32F4 boards, and this is the first release; offered for criticism, suggestions etc ...

(a) The RTS produced by AdaCore works to the bare metal when dealing with the board hardware. This seems to me to be a lot more work than necessary, given that the manufacturer provides a free-to-use BSP.

(b) I've included minimal interfaces to the board hardware: clock, buttons, LEDs. Given that the Ada RTS is minimal, these could have been put in their own library, not part of the RTS; but I was concerned that the board requires clock initialisation before the other hardware can be used, and the Ada RTS will require clock initialisation .. will probably revisit this decision.

(c) For tasking, I'm going to investigate FreeRTOS, which STMicroelectronics provide a copy of with their BSP. Does anyone know whether this is a completely stupid idea?

The release is at [1]: from the README,

This is an Ada Runtime System (RTS) for the GCC Ada compiler (GNAT), targeted to the STMicroelectronics STM32F429I Discovery board (see http://www.st.com/).

The RTS is a true zero-footprint system. Package System contains the following restrictions:

```
pragma Restrictions (No_Allocators);
pragma Restrictions (No_Delay);
pragma Restrictions (No_Dispatch);
pragma Restrictions
         (No_Enumeration_Maps);
pragma Restrictions
         (No_Exception_Propagation);
pragma Restrictions (No_Finalization);
pragma Restrictions
         (No_Implicit_Dynamic_Code);
pragma Restrictions (No_Protected_Types);
pragma Restrictions (No_Recursion);
pragma Restrictions
         (No_Secondary_Stack);
pragma Restrictions (No_Tasking);
```

The RTS contains object code for all the relevant drivers from STMicroelectronics' STM32Cube_FW_F4_V1.3.0 package, but not the source code. Makefile.inc (altered as necessary to match the place where you have installed the STM32Cube package) can be included in your own Makefiles to provide access to the drivers' header and source files; see the demonstration's Makefile.

The RTS has been built with no optimisation (-O0) and for debugging (-g), using GNAT GPL 2014 for arm-eabi-darwin-bin on Mac OS X (it should work out of the box with a Linux-hosted cross-compiler). Ada, C and C++ demo programs are included.

The RTS is intended to support commercial binary distributions[2]. The Ada source code has either been derived from FSF GCC (4.9.1) or written for this work; see the files COPYING3 and COPYING.RUNTIME. The C source has either been derived from STMCube or written for this work: see the file COPYING.STMicroelectronics.

[1] https://sourceforge.net/projects/stm32f4-gnat-rts/files/20141207/

From: Simon Wright <simon@pushface.org>
Date: Thu, 05 Feb 2015 11:26:44 +0000
Subject: ANN: STM32F4 GNAT Run Time Systems 20150204
Newsgroups: comp.lang.ada

This is the third release of a GNAT RTS with the GCC Runtime Library exception for STM32F4 boards.

(a) Tasking is implemented using FreeRTOS[3], which STMicroelectronics provide a copy of with their BSP.

(b) I've included minimal interfaces to the board hardware: clock, buttons, LEDs. Given that the Ada RTS is minimal, these could have been put in their own library, not part of the RTS; but I was concerned that the board requires clock initialisation before the other hardware can be used, and the Ada RTS requires clock initialisation .. will probably revisit this decision.

The release is at [4]: it contains two RTSs, one (in stm32f429i-disco-bsp; demonstrators in demo-stm32f429i-disco-bsp) has the barest minimum for Ada support, while the more interesting one (in stm32f429i-disco-rtos; demonstrators in demo-stm32f429i-disco-rtos) supports Ravenscar tasking, allocators, tagged types, and the secondary stack. From its README:

This is an Ada Runtime System (RTS) for the GCC Ada compiler (GNAT), targeted to the STMicroelectronics STM32F429I Discovery board (see http://www.st.com/).

The RTS supports Ravenscar tasking. Package System contains the following additional restrictions:

```
pragma Restrictions
         (No_Enumeration_Maps);
pragma Restrictions
         (No_Exception_Propagation);
pragma Restrictions (No_Finalization);
pragma Restrictions (No_Recursion);
```

The RTS contains object code for all the relevant drivers from STMicroelectronics' STM32Cube_FW_F4_V1.3.0 package, but not the source code. Makefile.inc (altered as necessary to match the place where you have installed the STM32Cube package) can be included in your own Makefiles to provide access to the drivers' header and source files; see the demonstration's Makefile.

The RTS has been built with no optimisation (-O0) and for debugging (-g), using GCC 4.9.1 for arm-eabi-darwin13-bin[1] on Mac OS X (it should work out of the box (but after

recompilation!) with a Linux-hosted cross-compiler.

The RTS is intended to support commercial binary distributions[2]. The Ada source code has either been derived from FSF GCC (4.9.1) or written for this work; see the files COPYING3 and COPYING.RUNTIME. The C source has either been derived from STMCube or written for this work: see the file COPYING.STMicroelectronics.

The RTS is based on FreeRTOS[3], as customised by STMicroelectronics in the STM32Cube package. See COPYING.FreeRTOS.

The following non-original files don't form part of a binary deliverable, so don't affect the status of the binary:

- build_runtime.gpr and runtime.xml originated in AdaCore's GNAT GPL 2014 arm-eabi distribution (for Linux).

- The linker script stm32f429i-flash.ld is under an MIT licence: see COPYING.MIT.

[1] https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/4.9.1bis/arm-eabi/

[2] STMicroelectronics' evaluation product licence agreement at www.st.com/epla forbids the sale of products including this board, so this work would have to be reconfigured for a different board anyway.

[3] http://www.freertos.org

[4] https://sourceforge.net/projects/stm32f4-gnat-rts/files/20150204/

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Thu, 05 Feb 2015 18:06:45 +0200
Subject: Re: ANN: STM32F4 GNAT Run Time Systems 20150204
Newsgroups: comp.lang.ada

> [...]

Nice to see people working actively on this area.

> [...]

Personally I would prefer BSD/ISC/MIT for all possible parts, like you have done for the linker script.

For example, see my STM32F4 package:

https://bitbucket.org/tkoskine/gnat-arm-app-skeleton/src/ab141a07860925a7c19d333c30b6084a041d9325/stm32f4.ads?at=default

> [...]

If someone has time, Ubuntu/Debian/Fedora packages would be nice. :)

For AVR-Ada, I have tried to provide unofficial packages[1,2] which can be installed with apt-get/yum and then hello world project works out of the box:

```
apt-get install avr-ada
cd hello-avr-ada
avr-gnatmake -XBOARD=arduino_uno hello
```

Similar procedure for ARM/STM32F4 should be doable (although takes some time).

> [2] STMicroelectronics' evaluation product licence agreement at www.st.com/epla forbids the sale of products including this board, so this work would have to be reconfigured for a different board anyway.

Olimex sells many different STM32Fx boards which don't have this restriction. I have used[3] STM32-P405 and STM32-E407[4] boards.

[1] http://ubuntu.ada-language.com/

[2] http://fedora.ada-language.com/

[3] http://tero.stronglytyped.org/running-ada-2012-on-olimex-stm32-e407-arm-cortex-m4-stm32f4.html

[4] https://www.olimex.com/Products/ARM/ST/STM32-E407/

Request: Virtual Analytical Engine

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Wed, 10 Dec 2014 07:33:04 -0500
Subject: Re: Happy Birthday Ada Lovelace
Newsgroups: comp.lang.ada, fr.comp.lang.ada

> [...]

Has anyone ever attempted to create a software simulation of the Analytical Engine? It would be great if we could actually execute the Lady Lovelace's programs, even if only in a virtual machine. Why do I have the feeling she would have appreciated the elegance of that?

P.S. Of course the Analytical Engine simulation should be written in Ada!

Qt5Ada

From: Leonid Dulman <leonid.dulman@gmail.com>
Date: Wed, 10 Dec 2014 20:36:15 -0800
Subject: Announce : Qt5Ada version 5.4.0 (387 packages) and VTKAda version 6.1.0 (656 packages) release 10/12/2014 free edition
Newsgroups: comp.lang.ada

Qt5Ada is an Ada 2012 binding to the Qt5 (5.4.0-final) framework. The C binaries were built with Microsoft Visual Studio 2012 on Windows and with GCC x86-64 on Linux and Mac OS X.

The package was tested with GNAT-GPL-2012 on 32 and 64 bit Windows, on Debian/Linux x86-64 (7.3) and Mac OS X (10.8.5).

It supports GUI, SQL, multimedia, web, network, touch devices, sensors and many others things.

Added QtOpenGL support (QtOpenGLWindow, QtOpenGLWidget, QtOpenGLFunction and others), new packages and demos.

Qt5Ada for Windows and Linux (Unix) is available from

http://users1.jabry.com/adastudio/index.html

My configuration script to build Qt 5.4 is: configure -opensource -release -nomake tests -opengl desktop -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/5.3"

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

I do this work on my own risk (after 9-12 hours at the factory) and I hope Qt5Ada and VTKAda will be useful for students, engineers, scientists and enthusiasts.

With Qt5Ada you can build any applications and solve any problems easy and quickly.

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

[See also “Qt5Ada”, AUJ 35-3, p. 158. —sparre]

Muen Separation Kernel

From: Adrian-Ken Rueegsegger <ken@codelabs.ch>
Date: Mon, 12 Jan 2015 20:34:21 +0100
Subject: [ANN] Muen development version 0.6 released
URL: https://groups.google.com/forum/#!topic/muen-dev/_HM6w9toM-Y

We are proud to announce the availability of Muen version 0.6, which marks the first official development release.

The following major features and improvements have been implemented since the last announcement:

- Migration of Muen to SPARK 2014

The kernel code has been migrated from SPARK 2005 to SPARK 2014 [1]. Absence of runtime errors is now verified using the GNATprove tool. Switching to SPARK 2014 enables the use of a larger Ada language subset and contracts are expressed as Ada 2012 aspects. Replacing the SPARK 2005 annotations with Ada 2012 contracts made the code much cleaner.

- PCI device passthrough using Intel VT-d (DMAR and IR)

Hardware passthrough is realised using Intel's VT-d DMA and interrupt remapping technology. This enables the secure assignment of PCI devices to subjects.

- XML policy abstraction and enhanced tool support

The XML system description has been modularised and additional abstractions have been added to the policy. This enables users to integrate complex component-based systems running on top of the Muen kernel.

Further changes and improvements include:

- Support for Intel Haswell architecture
- Lightweight subject timer mechanism
- Scheduler improvements (minor frame synchronisation)
- Subject Monitor migrated to SPARK 2014
- Debug server subject

Despite the addition of all these new features the kernel has retained its small size. Some numbers regarding the size of Muen: the kernel including the minimal zero-footprint runtime consists of a total of 5308 source lines of code (SPARK/Ada: 4979, Assembly: 329, as reported by SLOCCount version 2.26).

A high-level document describing the process of configuring and building a component-based system running on the Muen Separation Kernel is available on the project website [2].

We would also like to mention that we gave a talk about Muen at the High Integrity Software conference HIS 2014 [3] in Bristol. The slides are available online at [4].

Further information is available on the project website [5] and the git repository is at [6].

Please feel free to give the development version of Muen a try. Feedback is much appreciated!

[1] http://spark-2014.org/

[2] http://muen.codelabs.ch/muen-toolchain.pdf

[3] http://www.his-2014.co.uk/

[4] http://www.slideshare.net/AdaCore/slides-his-2014secunethsr

[5] http://muen.codelabs.ch/

[6] http://git.codelabs.ch/?p=muen.git

[See also “Muen Separation Kernel”, AUJ 35-1, p. 14. —sparre]

Ada-related Products

Janus/Ada

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 Nov 2014 16:47:21 -0600
Subject: Re: What exactly is the licensing situation with GNAT?
Newsgroups: comp.lang.ada

> [...]

You could of course use a different, commercial Ada compiler, rather than insisting on GNAT. At least Janus/Ada still costs $195 for the personal version and $500 for the professional version. See www.rrsoftware.com. (Disclaimer for new people here, I'm a co-founder and primary author of Janus/Ada, so I'm a bit biased. :-)

> [...] what is missing is some sort of intermediate license for people who just want to write small scale applications and don't have the security requirements of big projects and thus don't need the support that AdaCore offers. Something like Turbo Pascal in the past or maybe even Visual Studio in the Personal Edition or so.

Aonix used to have an ObjectAda version like that, but I heard that they got rid of it as they couldn't afford to support it. Janus/Ada is in that price range as well, but I have to admit the same is true -- there isn't enough business to justify working on it full-time. I have to do standardisation stuff and ACATS stuff to make ends meet.

[...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 Nov 2014 20:10:39 -0600
Subject: Re: What exactly is the licensing situation with GNAT?
Newsgroups: comp.lang.ada

> [...]

I've been holding off on updating the website until all of the Windows 7 issues are cleared.

> What is the status of Janus Ada and CLAW with regard to Windows 7 & 8?

Windows 8 == Vista to me; I'm waiting for Windows 10 as a system that tried to make the desktop a second-class citizen is near-worthless for programming. So no testing there. (Tom Moran reported that everything works the same there as it does on Windows 7; I don't think there are any additional issues, but as I said, I didn't try them.)

As far as Windows 7 goes, there are a few minor problems that have so far kept the compiler in beta. The main one is that the uninstaller doesn't work (I can't seem to convince Windows that it should have the permissions to uninstall, even when run explicitly as an administrator). The minor one (considering that it is obsolete even if it works) is that the GUI programming environment doesn't work at all; you'd have to use some other editor. (Most people prefer to do that anyway, but it really ought to work; to do that, it will have to be totally replaced, something I don't have time for right now.)

The Claw binding and Claw programs work on Windows 7, so far as I can tell via testing. The Claw Builder comes up with a white screen for some reason on Windows 7, so it isn't usable right now on that system. (I'm guessing that there is a deadlock situation in the way Claw writes the overlay, but it will take some intensive testing to figure out the cause.) A more minor problem is that the help files have to be found manually every time you open them, for some reason Windows 7 can't remember where they are.

Anyway, the beta works on Windows 7, with some glitches. Once the glitches are gone, I'll update the web site, too.

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Thu, 13 Nov 2014 18:51:32 +0200
Subject: Re: What exactly is the licensing situation with GNAT?
Newsgroups: comp.lang.ada

> [...]

Janus/Ada works on 64-bit Windows 8.1. There are some quirks with filenames (they seem to be partially case sensitive), but otherwise everything is ok. (I regularly test my Ada software on Windows 8.1 with Janus/Ada.)

On 64-bit Windows 7 everything works fine.

http://build.ada-language.com/view/JanusAda/ provides build logs for some of my Ada packages compiled on Windows 7.

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Sun, 23 Nov 2014 18:33:09 +0200
Subject: Re: What is the situation with Janus Ada?
Newsgroups: comp.lang.ada

> [...]

Using Janus/Ada with WINE works. I semi-regularly test my Ahven library with Janus/Ada and WINE on Fedora Linux 20.

Mingw is not required, but Microsoft Windows SDK is (the one which contains the linker and some libraries).

I am currently using Janus/Ada on 32-bit Windows XP, 64-bit Windows 7, and 64-bit Windows 8.1. (WINE is set to use 32-bit mode.)

Ada and Operating Systems

Mac OS X: GNAT GPL 2014 for ARM-EABI

From: Simon Wright <simon@pushface.org>
Date: Wed, 19 Nov 2014 09:54:06 +0000
Subject: ANN: GNAT GPL 2014 for arm-eabi on Darwin
Newsgroups: comp.lang.ada

Find this at:
https://sourceforge.net/projects/gnuada/fil

es/GNAT_GPL%20Mac%20OS%20X/2014-arm-eabi-darwin-bin/

This is GNAT GPL 2014, rebuilt as a cross-compiler from Mac OS X to arm-eabi. Runtimes for two STM32F4 boards, and examples, are included:

- STM32F4 Discovery

- STM32F429I Discovery

The compiler is known to run on Mavericks and Yosemite.

For installation, untar gnat-gpl-2014-arm-eabi-darwin-bin.tar.gz, enter gnat-gpl-2014-arm-eabi-darwin-bin/ (there is a README) and run doinstall. Note that you must have a working host compiler; if using Mavericks, this should be GNAT GPL 2014, but if using Yosemite (at the time of writing, 17.xi.2014, on which GNAT GPL 2014 compilations fail) you can use a compiler with an equivalent set of tools, say FSF GCC 4.9.1 from [1].

Additionally, stlink-darwin-bin.zip contains a .tar.gz file with the stlink utilities used to communicate with the boards over USB, and a README which details installation.

Usage notes are in the AdaCore "GNAT Pro User's Guide Supplement for Cross Platforms"[2], specifically in section K.2[3]. Note however that that document is a work-in-progresss and discusses features that didn't make it into the GNAT GPL 2014 release:

o gprbuild doesn't support the Project attribute Runtime: instead, in package Builder, Default_Switches ("Ada") should include "--RTS={runtime}".

[1] https://sourceforge.net/projects/gnuada/

[2] http://docs.adacore.com/gnat_ugx-docs/html/gnat_ugx.html

[3] http://docs.adacore.com/gnat_ugx-docs/html/gnat_ugx_14.html#SEC204

[See also “GNAT for More ARM Variants”, AUJ 35-4, p. 217. —sparre]

Windows: Gnoga

From: David Botton <david@botton.com>
Date: Thu, 27 Nov 2014 20:56:37 -0800
Subject: Gnoga on MinGW 32 and 64bits
Newsgroups: comp.lang.ada

It is now possible to build and use the current distros of MinGW (gcc 4.8.1) 32 and 64 bits for building and using Gnoga since AWS is no longer required. So it is now possible to build unencumbered (i.e. no GPL virus) versions of Gnoga apps on Windows 32 and 64.

Since MinGW does not include gprtools I have added make.bat and clean.bat which use gnatmake -P instead.

If you use MinGW under Cygwin you will need to change from gnatmake to x86_64-mingw32-gnatmake (for 64bits) or i686-w64-ming32-gnatmake (for 32bits)

Mac OS X: Native GUIs with Gnoga

From: David Botton <david@botton.com>
Date: Thu, 18 Dec 2014 11:11:36 -0500
Subject: Gnoga native Mac OS X application support added
Newsgroups: gmane.comp.lang.ada.macosx

To: GNAT-OSX-dhAwdhUhaNgMT+7pcfOT8A@public.gmane.org

http://www.gnoga.com

See docs/native_mac_apps.md for more information, but here is a summary:

1. Create a singleton app using Gnoga

2. Make native support for Mac using:

```
make native_osx
```

3. Copy your project's individual bin, js, etc. directories to deps/MacGap2/public

4. Modify the index.html file in deps/MacGap2/public to contain the following lines:

```
<script type="text/javascript"
        charset="utf-8">

var p = MacGap.resourcePath +
"/public/bin/YOUR_GNOGA_APP_NAME";

MacGap.launch (p);

window.open("http://127.0.0.1:8080","_self")

</script>
```

Note: The index.html page can be used to display some sort of "loading" message if desired.

5. From the deps directory run - open MacGap2/MG.xcodeproj/

6. Build as you would any native Mac OS X application for XCode.

Windows: ObjectAda Special Edition

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Fri, 23 Jan 2015 02:37:07 -0800
Subject: Re: Free Downloadable Ada95 Compilers
Newsgroups: comp.lang.ada

> Are there any free downloadable Ada95 compiler except the GNAT one? [...]

Here, the ObjectAda 7.2.2 Special Edition

http://www.ada-deutschland.de/sites/default/files/AdaTourCD/AdaTourCD2004/entwicklungsumgebung/Software/ObjectAda7.22/zip.zip

(from the page: http://www.ada-deutschland.de/sites/default/files/AdaTourCD/AdaTourCD2004/index_tools.html)

Mac OS X: GNAT

From: Simon Wright <simon@pushface.org>
Date: Sun, 25 Jan 2015 16:41:43 +0000
Subject: ANN: gcc 4.9.1bis for Darwin
Newsgroups: comp.lang.ada

This is to announce two GCC 4.9.1 compilers, one a Darwin native compiler (the same as previously uploaded, but can be installed in a place of your choice) and one a cross-compiler to arm-eabi, aka arm-none-eabi, as found on the STMicroelectronics[1] STM32F4 series.

Both compilers work on Mavericks and Yosemite.

The compilers are at the usual place[2]. They each have a similar installation mechanism as that in the GNAT-GPL series, so you can choose where to install them (the default is /opt/gcc-4.9.1, but /usr/local/gcc-4.9.1 works too; there may be problems with longer paths). You can install the cross compiler on top of the native one.

The cross-compiler comes without an RTS. You can find suitable RTS at [3], together with a compiled copy of stlink (the tools that enable download to the board and debug). The 20150124 version comes in two variants: one that just supports the STCube BSP, and - more interestingly - one that additionally supports Ravenscar tasking via FreeRTOS[4].

The tasking RTS has the following restrictions (aside from pragma Profile (Ravenscar)):

```
pragma Restrictions (No_Allocators);
pragma Restrictions (No_Dispatch);
pragma Restrictions
       (No_Enumeration_Maps);
pragma Restrictions
       (No_Exception_Propagation);
pragma Restrictions (No_Finalization);
pragma Restrictions (No_Recursion);
pragma Restrictions
       (No_Secondary_Stack);
```

and the following bugs/features (see the Tickets tab at [3]):

- You have to start tasking by calling FreeRTOS.Tasks.Start_Scheduler from your main program (it doesn't return unless something is horribly wrong).

- Ada.Real_Time.Clock is only valid for 50 days (and has a tick of 1 ms).

- The Interrupt_Priority aspect on a PO doesn't affect the actual interrupt's priority (it does affect the PO's ceiling priority).

- Some weird interaction between the compiler and the RTS code means that a protected spec hides package Interfaces. You can 'use Interfaces;' before the protected spec, though.

[1] http://www.st.com

[2] http://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/4.9.1bis/

[3] http://sourceforge.net/projects/stm32f4-gnat-rts/files/

[4] http://www.freertos.org

[See also “Mac OS X: GNAT”, AUJ 35-3, p. 160. —sparre]

References to Publications

Tutorial: Arduino Due (ARM Cortex-Mx)

From: Maciej Sobczak <maciej@msobczak.com>
Date: Thu, 29 Jan 2015 15:16:01 -0800
Subject: Ada on Cortex-M: tutorial for Arduino Due
Newsgroups: comp.lang.ada

I am pleased to announce the new tutorial for Ada on ARM Cortex-Mx, with examples for the Arduino Due board:

http://www.inspirel.com/articles/Ada_On_Cortex.html

This tutorial is intended for Ada beginners, but at the same time tries to present the bare-metal approach to embedded programming. It is not based on the existing STM32FxDiscovery packages and so is likely to propose some alternative ideas.

The tutorial is a work in progress and is intended to evolve with time. Your comments, including critical ones, are highly welcome.

Ada Inside

Rosetta/Philae

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 12 Nov 2014 17:36:38 +0100
Subject: PR while it's hot: Rosetta/Philae is in Ada
Newsgroups: comp.lang.ada

An extraordinary result of the Ada/HOOD combination! For the justification of this choice, see:

http://adsabs.harvard.edu/full/1997ESASP.409..133D

And tell your friends (an enemies too! ;-)

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 13 Nov 2014 12:23:02 +0100
Subject: Re: PR while it's hot: Rosetta/Philae is in Ada
Newsgroups: comp.lang.ada

> Is there any Ada using Hood tutorial or website.

You can start with http://en.wikipedia.org/wiki/HOOD_method

There is also the "HOOD book": http://www.adalog.fr/hoodbook.htm

There is also some information about HOOD on the ESA site.

For products supporting HOOD, see STOOD and CP-Hood on Ellidiss site (http://www.ellidiss.com)

Vermont CubeSat

From: Jonathan <johnscpg@googlemail.com>
Date: Wed, 12 Nov 2014 10:49:06 -0800
Subject: Re: PR while it's hot: Rosetta/Philae is in Ada
Newsgroups: comp.lang.ada

While we're at it, we should congratulate Peter Chapin and company for a successful CubeSat mission. I notice that the Wikipedia article says:

"Vermont Lunar is the only non NASA/Air Force CubeSat from this ELaNa IV launch that is fully working. Eight were never heard from at all."

http://en.wikipedia.org/wiki/Vermont_Lunar_CubeSat

http://embedded-computing.com/articles/2014-vermont-technical-college/#

http://www.cubesatlab.org/

Mine Detector

From: David Botton <david@botton.com>
Date: Thu, 4 Dec 2014 23:54:05 -0500
Subject: New Gnoga Demo
Newsgroups: gmane.comp.lang.ada.gnoga

Jeff Carter has sent me a copy of his Mine Detector game using Gnoga that is now multi connect. I've put it up as a demo at http://gnoga.com:8081

Web Chat with Gnoga

From: David Botton <david@botton.com>
Date: Fri, 9 Jan 2015 11:11:23 -0500
Subject: Gnoga Chat Demo from Jeff Carter
Newsgroups: gmane.comp.lang.ada.gnoga

To: Gnoga support list <Gnoga-list-5NWGOfrQmneRv+LV9MX5uipxlwaOVQ5f@public.gmane.org>

I put up a new demo from Jeff Carter (and it is also in git at demo/ chattanooga) on the Gnoga website, a chat app. I put it under https (although certificate is a test one) at:

https://chat.gnoga.com

Ada in Context

Machine_Overflows

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 23 May 2014 16:39:54 -0500
Subject: Re: How to check a Float for NaN
Newsgroups: comp.lang.ada

> [...]

[...] I have no idea what's supposed to happen if Machine_Overflows is False. To the point that Janus/Ada simply doesn't use it; Machine_Overflows is True and we check after every group of operations to ensure that's true. So it's not possible to generate a NaN in Janus/Ada. I understand that the other semantics exists, but it's a mess by any standard and I'd need a lot more convincing ($$$$) to support something else. [That is, I see no sensible reason for NaNs or infinities -- they're just ways of deferring detection of bugs. I would have hoped that Ada's moved beyond that, just like it has for integers.]

From: Maurizio Tomasi <ziotom78@gmail.com>
Date: Tue, 27 May 2014 05:35:26 -0700
Subject: Re: How to check a Float for NaN
Newsgroups: comp.lang.ada

> [...] I see no sensible reason for NaNs or infinities -- they're just ways of deferring detection of bugs. [...]

Being a scientist working with large chunks of data, I find NaNs useful in a number of situations. I work in a domain (observational cosmology) where we need to deal with sky maps containing ~10^7 pixels (you can think of a "map" as a 1D vector where pixels on the sky sphere are ordered according to some rule). Not every sky direction can be sampled, because of a number of problems (in the instrument, in the observational strategy, in the data reduction pipeline, etc.)

Therefore, in my Python+NumPy codes I always mark such directions using "quiet NaNs". If I have to combine two maps in order e.g. to take their average, the usual rules for combining NaNs are be exactly what I want. Writing in Ada what I actually write in Python:

```
for I in Map1'Range loop
   Average_Map (I) :=
         0.5 * (Map1 (I) + Map2 (I));
end loop;
```

If either Map1(I) or Map2(I) (or both) are NaN, then Average_Map(I) will be a NaN too, which is correct from the point of view of the meaning of the measurement. But without proper treatment of NaNs, one should write:

```
for I in Map1'Range loop
   if Is_NaN (Map1 (I)) or Is_NaN (Map2 (I))
   then
      Set_To_NaN (Average_Map (I));
   else
      Average_Map (I) :=
         0.5 * (Map1 (I) + Map2 (I));
   end if;
end loop;
```

If one has to run many calculations on such maps (which is indeed always the case) instead of just a plain average, the code can get quite complex. And I do not think one gets more safety from such

verbosity, as what a scientist expects from a NaN number is actually what the usual rules for NaN give.

I am not an Ada expert, so these are just my two cents.

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 27 May 2014 08:53:00 -0700
Subject: Re: How to check a Float for NaN
Newsgroups: comp.lang.ada

[...]

But you don't need NaN's built into the language in order to get that sort of functionality. In Ada (or C++ or any other language that supports operator overloading), it's simple enough to define an "optional floating-point" record type consisting of a float and a Boolean, where the Boolean indicates "missing data", and define operators that produce "missing data" if either operand is missing. So you could still write mostly the same code, except that converting to or from a float, or from a floating-point literal, takes a little extra code.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 27 May 2014 17:35:48 -0500
Subject: Re: How to check a Float for NaN
Newsgroups: comp.lang.ada

> [...]

If one has

```
function "+" (Right : Float) return
        Optional_Float;
```

then the "extra code" is just preceeding the float value with a "+". Hardly earth-shaking. (And the usual complaint about using "+" as a conversion operator is a non-problem here as these are numeric types).

I much prefer this sort of solution (where the missing values are explicitly treated) rather than using some sort of magic number (a NaN being an extreme version of that). The name alone tells you that it doesn't belong in a numeric type -- since when is something that is "not a number" belong in a type defining numbers?

As usual, this is mainly a case of premature optimisation (perverting the hardware to handle something that's a rare need -- I wonder how much faster float hardware could be if it didn't have to mess with NaNs? I know that they impacted our software floating point quite a bit even though I made no attempt to actually do anything useful with them.)

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Tue, 27 May 2014 15:59:07 -0700
Subject: Re: How to check a Float for NaN
Newsgroups: comp.lang.ada

> [...] using some sort of magic number (a NaN being an extreme version of that) [...]

It's a clear violation of the software-engineering principle that a value has only a single meaning. (Of course, returning zero from Ada.Strings.Fixed.Index is the same error.)

Termination of Periodic Tasks

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 17 Jun 2014 16:51:07 +0200
Subject: Re: Termination of periodic tasks
Newsgroups: comp.lang.ada

> The RM is silent about the order in which tasks are awaited. [...]

There is no order, because all tasks terminate together (9.3 (6..9)).

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 17 Jun 2014 15:00:52 -0500
Subject: Re: Termination of periodic tasks
Newsgroups: comp.lang.ada

> Thus per design there is no way to make a non-trivial library-level task to complete without means outside the library level. [...]

Actually, there is a way, at least if you want to ensure that the program cleans itself up properly. We invented it for Claw, and I put it into the ACATS so it's pretty certain that all compilers support it.

The trick is to use Ada.Task_Identification to find out whether the environment task is trying to exit.

```
if not Is_Callable (Environment_Task) then
    return; -- Exit this task.
end if;
```

Is_Callable will only be False for the environment task if the main subprogram has exited and we're waiting for library-level tasks to complete. In that case, we want to kill off this task. (Note: Not all Ada 95 compilers did this at the time, some always returned true from it no matter what. But that would fail ACATS test CXC7004 in modern compilers, so it's unlikely that many get this wrong. One might want to look at that ACATS test for a complete example of the method.)

It can be clunky to get this into the task somewhere; it works best if the task is actively polling (as the message loop task in Claw is always doing).

Note: function Environment_Task was added to the package in Ada 2012. For earlier Ada, one needs to have the elaboration of the package containing the task squirrel away the task id:

```
Environment_Task_Id : constant Task_Id :=
        Current_Task;
```

From: Charles H. Sampson <csampson@inetworld.net>
Date: Tue, 17 Jun 2014 13:14:23 -0700
Subject: Re: Termination of periodic tasks
Newsgroups: comp.lang.ada

> > Why not have a "stop" entry called by the main program when it terminates (possibly through an exported subprogram if you don't want to have the task public)?

> Mostly because I believe this to be too heavy for a burden for the client, and somewhat of an abstraction leak.

I'm having trouble understanding how this is too heavy. In most programs I've written, there's a possibility of stopping. Usually the need to stop is detected by or, most commonly, propagated to the main program. The main program then signals all of its library packages to do whatever is necessary for stopping. An exported Stop subprogram seems a quite natural way to do that.

I've even used implementations that have exported Stop subprograms in all library packages, some of them null. That enforces the abstractions in that the main program doesn't need to know which library packages need to be wrapped up.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 18 Jun 2014 09:32:23 +0200
Subject: Re: Termination of periodic tasks
Newsgroups: comp.lang.ada

[...]

> I've even used implementations that have exported Stop subprograms in all library packages, some of them null. [...]

I prefer a stateless design of packages with explicit objects maintaining the state, created by the client. That eliminates the problem of task termination too.

Parsing Ada Source Text

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Thu, 19 Jun 2014 17:13:19 -0400
Subject: Re: Ada platforms and pricing, was: Re: a new language, designed for safety !
Newsgroups: comp.lang.ada

> My understand is that parsing Ada requires name resolution to resolve syntactic ambiguities.

Yes. The syntax rules given in the RM are ambiguous. For example, in a context expecting an expression, "X(Y)" could be a function_call, type_conversion, indexed_component, or slice. (Did I forget any?) Likewise, in statement context, that same text could be a procedure_call or an entry_call.

The "X" in "X.Y" could be a name or an implicit_dereference.

But...

>...This means symbol table management and dealing with Ada's visibility rules has to be done while parsing is taking place.

But no, it doesn't mean that, and in fact mixing semantic analysis with parsing is highly undesirable. The parser should build a tree, and not any "symbol table" kinds of things. The output of the parser should depend ONLY on the contents of a single source file; it shouldn't need to know about separate compilation.

Semantic analysis then walks the tree built by the parser.

The way to deal with an expression "X(Y)" is for the parser to build a tree node that represents "something that looks like a call or a type_conv or ...". That is, "a name followed by a parenthesized, comma-separated sequence of expressions". That has been called an "Apply" node in some compilers.

Basically, you need to write a grammar for Ada that is unambiguous, and that allows a superset of what the RM grammar allows.

When semantic analysis sees an Apply node for X(Y), it looks up X and Y. It might find X denotes a type, or denotes one or more functions, or ...

I've done serious work on about 7 or 8 Ada compilers, and this is how ALL of them worked. (Here, I'm counting independently designed compiler front ends, not different host/target platforms. That is, GNAT counts as "1 Ada Compiler" I've worked on, even though it supports many platforms.)

[...]

C compilers typically work the other way (mixing parsing with semantic analysis), to solve syntactic ambiguities related to "typedef". IMHO any language that forces that design on a compiler is broken. I'm not sure C forces that design; maybe the typedef problem could be solved differently, but it doesn't look easy to me.

The Main Features of Ada

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Mon, 23 Jun 2014 09:18:11 +0300
Subject: Re: Ada platforms and pricing, was: Re: a new language, designed for safety !
Newsgroups: comp.lang.ada

> [...]

While I would like to have the standard Ada tasking, timing, and exception support, I would most definitely prefer an Ada subset without them, over C. As I remember, prof. McCormick's experience from the model-railway exercise (where his students failed when they used C, but succeeded with Ada) identified Ada's advantage over C to be in the better scalar typing, not in the features that require a full RTL (but I don't remember if exceptions were a factor in McCormick's experience).

I am currently working on an Ada project with a null run-time and a proprietary small multi-threading kernel. But I still feel I am "doing Ada", although with a different syntax for tasking. All the advantages of the type and package system are still there, and they are, to me, the main feature of Ada (in this application domain, at least).

Avoiding Exceptions

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 25 Nov 2014 16:12:07 -0600
Subject: Re: How to get nice with GNAT?
Newsgroups: comp.lang.ada

Åke Ragnar Dahlgren wrote:

> Of course I always listen seriously to Jeff Carter but it's not obvious to me that doing "C in Ada" is bad.

It's bad. :-)

> If I remember correctly Google employees are recommended to avoid using exceptions when doing C++. The designers of Google Go has gone great lengths to avoid the exception concept as much as possible.

Very bad advice, IMHO. With one exception (pun intended):

> In addition SPARK forbids usage of exceptions.

While I think SPARK would be better served with limited exception support, at least they require a proof that no exceptions can be raised.

The reason I feel so strongly about this is that exceptions (especially Constraint_Error and Program_Error) point out bugs in your code. Whenever you "eat" an exception (turning it into an error code, or simply ignoring it), you've put an opportunity to ignore a bug into your code. With all of the potential problems that entails.

To take a concrete example. My web server runs with all exceptions enabled, and there is very little handling of exceptions (there are a few cases where expected exceptions are handled, as when a TCP/IP connection is unexpectedly dropped). Mainly, the worker tasks handle any surprise exceptions, log them, and reset everything in that task to a fresh state. Doing this prevents most bugs from causing security problems -- while a crafted input might cause one worker to fail, that only causes the sender to get no response. Other connections (workers) are unaffected, and there is almost no chance of a detected bug from overwriting memory or disk or any of the other things that cause security problems.

Exceptions surely aren't enough to prevent all security issues, but they can help avoid a substantial number of them.

(As previously noted, if you could prove that no exceptions are possible - meaning that no low-level bugs are possible - that would be better than having to figure out last-chance handlers and the like, but that's still beyond the state of the art for general purpose code. When that changes, I'll reconsider my stance on exceptions, but not until then.)

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 26 Nov 2014 14:06:24 +0100
Subject: Re: How to get nice with GNAT?
Newsgroups: comp.lang.ada

> [...]

If exceptions were under a contract, the list of possible exceptions to catch would be definite and quite small in most cases. So, actually, we could disallow "when others" for all subprograms having an exception contract and calling only such subprograms (statically).

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 26 Nov 2014 15:27:35 -0600
Subject: Re: How to get nice with GNAT?
Newsgroups: comp.lang.ada

> [...]

The practical problem with exception contracts is that they encourage people (especially coders rather than engineers) to "eat" exceptions rather than to figure out what ought to be done with them (or better yet, redoing the code/preconditions/predicates so they can't arise). That's the practical experience with them in Java, and that has caused some ARG members to be rather strongly against them. (Which is why they didn't make it into Ada 2012.)

I personally find that misguided (because Ada is for engineers, not coders), and I'll try again with them the next Ada amendment.

BTW, that's a problem with all statically categorisation contracts. We're looking at a potentially blocking categorisation as part of the parallelisation effort. What happens there is that calling any routine that is potentially blocking is illegal inside of a routine that is declared as non-blocking. (And unlike exception contracts, there's no workaround). The effect is that one has to change the status of lots of routines in order to use the categorisation. (At least for this particular categorisation, Ada already says which language-defined routines are potentially blocking, so it's just a matter of putting that into aspects and pragmas as needed - no arguments about whether Sin should be potentially blocking :-)

Even so, I think statically checked contracts and categorisations are going to be important, because they eliminate bugs at the source (and thus eliminate the need to worry about how to handle a substantial proportion of errors).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 27 Nov 2014 09:52:36 +0100
Subject: Re: How to get nice with GNAT?
Newsgroups: comp.lang.ada

> [...] (Which is why they didn't make it into Ada 2012.)

Right, but only if you require all subprograms to have contracts. Why should we? We couldn't anyway because it would break backward compatibility. Thus, IMO, there is nothing to worry about here.

> [...] I'll try again with them the next Ada amendment.

Good

> [...] The effect is that one has to change the status of lots of routines in order to use the categorisation.

Which is desired, isn't it?

> [...]

Conditionally blocking? Some predicates could depend (statically) on expressions. This is important for exceptions as well:

```
generic
   with procedure Visitor (E : Element);
procedure Iterate (X : Container);
```

If Visitor is not contracted and Iterate is, then the contract of Iterate should be "I raise, what Visitor does".

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 1 Dec 2014 16:25:28 -0600
Subject: Re: How to get nice with GNAT?
Newsgroups: comp.lang.ada

Dmitry A. Kazakov wrote:

> The idea that all/most/some bugs should somehow manifest their wrong behaviour in exceptions is dubious.

Fascinating. I'd say the reverse: that almost all bugs quickly manifest themselves in an exception (at least in well-designed Ada code). For instance, I tend to make off-by-one errors in index calculations. Such errors almost always result in a Constraint_Error when the index is used. Similarly, in Janus/Ada, we've sometimes passed the wrong entity to a subprogram; that almost always shows up as a Constraint_Error detecting the use of a non-existent variant. (If a routine expects a symbol table pointer to an object, and gets a package, the components it needs aren't going to be there.)

Indeed, the recent history of Ada includes more and more ways to specify what is expected/needed for a parameter/object/component. Null exclusions (Ada 2005), preconditions, and predicates (Ada 2012) are all ways to more closely tell the compiler what is intended.

The next step, IMHO, is to include exception contracts that effectively require exceptions not to occur. If they in fact do occur, then the program is wrong and will be rejected by the compiler. That means that "unexpected" Constraint_Errors will be detected statically and thus the manifestation of many bugs can be detected -- thus eliminating the bugs at the source.

Of course, once that next step is taken (and I mean in the context of the full Ada language, not just some simple subset like SPARK), then you'll probably be right. But that's still some distance in the future.

Subprograms outside packages?

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Sun, 11 Jan 2015 14:51:24 -0500
Subject: Re: Multiple procedures in the same adb file?
Newsgroups: comp.lang.ada

[...] It is unwise to have procedures outside of packages [...]

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Sun, 11 Jan 2015 23:38:34 +0200
Subject: Re: Multiple procedures in the same adb file?
Newsgroups: comp.lang.ada

> [...]

Why "unwise"? I agree it is unusual, but I find it is sometimes useful, in particular to have subprograms which are children of packages but are their own compilation units. In a layered architecture, such subprograms are sort of in a layer between the higher layer that contains the declaration of the parent package, and the lower layer that contains the body of that package.

In language-lawyer terms, perhaps such subprograms are not really "outside of packages", because child units are in some sense "inside" their parents, but the child subprograms are not "inside" any package in terms of source-code files.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Sun, 11 Jan 2015 19:53:17 -0500
Subject: Re: Multiple procedures in the same adb file?
Newsgroups: comp.lang.ada

> Why "unwise"?

Because sometimes you need to add new stuff (another related procedure, for example), and then there's no good place to put it. If the existing procedure is in a package, then that's where you put it.

This happened to me: Existing code:

```
function Some_Root_Package.
      Some_Function (...) return String;
```

Now I happen to know that it always returns String (1..20). And returning known-length strings is more efficient, and this one is a bottleneck, so it's worth changing to something like:

```
subtype String_20 is String (1 .. 20);
function Some_Function (...)
      return String_20;
```

But that doesn't work without adding a new package, which breaks compatibility. And this was a widely-used library, so I couldn't do that. And String_20 really doesn't belong in Some_Root_Package, nor anywhere else than the package that Some_Function is (directly) in (which didn't exist!).

If the original programmer had put Some_Function in a child package Some_Root_Package.Some_Package in the first place, then I wouldn't have had these problems.

I admit: these concerns are only significant for widely-used libraries. If I had been working on a program instead of a library, I would have just moved Some_Function into a package, and modified all the call sites. But it's probably a good idea to get in the habit of using "library-programming" style even when doing "program-programming", if it's not too much trouble.

Besides, it complicates the language quite a bit to have misc stuff allowed as compilation units (e.g. task body subunits). I would prefer that the only unit of separate compilation be the package (and maybe generic package).

> ... I agree it is unusual, but I find it is sometimes useful, in particular to have subprograms which are children of packages but are their own compilation units. In a layered architecture, such subprograms are sort of in a layer between the higher layer that contains the declaration of the parent package, and the lower layer that contains the body of that package.

Sure, but you can do that with another package just as well as with a procedure.

> In language-lawyer terms, perhaps such subprograms are not really "outside of packages", [...]

Yes, you're right. In fact, everything is logically nested within package Standard, so everything is in a package. But I meant that everything should be physically/textually nested within a package, and packages (and generic packages) should be the only separately-compiled things. And main procedures should be (textually) in packages, instead of as standalone procedures (and that part is not Ada -- just my preference for how it ought to be!).

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Mon, 12 Jan 2015 10:01:28 +0100
Subject: Re: Multiple procedures in the same adb file?
Newsgroups: comp.lang.ada

> This happened to me: [...]

Put the function in a package, and declare Some_Function as:

```
function Some_Function (...) renames
        New_Package.Some_Function;
```

(For compatibility, new programs would use directly the function in the package.)

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Mon, 12 Jan 2015 09:57:58 -0500
Subject: Re: Multiple procedures in the same adb file?
Newsgroups: comp.lang.ada

> [...]

I started to reply, "Good idea", but then I remembered it doesn't work. A library unit can only rename another library unit. If it did work, I'd still prefer to use packages from the start.

I see AARM-10.1.1(14.a) explains why we have this restriction, in part "because they wouldn't be particularly useful". Probably written by somebody who doesn't like library subprograms. ;-).

Conference Calendar

Dirk Craeynest
KU Leuven. 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.

2015

April 11-18 — 18th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2015), London, UK. Events include: CC (International Conference on Compiler Construction), ESOP (European Symposium on Programming), FASE (Fundamental Approaches to Software Engineering), FOSSACS (Foundations of Software Science and Computation Structures), POST (Principles of Security and Trust), TACAS (Tools and Algorithms for the Construction and Analysis of Systems).

- April 12 — 12th **International Workshop on Formal Engineering approaches to Software Components and Architectures** (FESCA'2015). Topics include: modelling formalisms, temporal properties and their formal verification, interface compliance and contractual use of components, static and dynamic analysis, industrial case studies and experience reports, etc.

April 12-15 — 23rd **High Performance Computing Symposium** (HPC'2015), Alexandria, VA, USA. Topics include: high performance/large scale application case studies, multicore and many-core computing, distributed computing, tools and environments for coupling parallel codes, high performance software tools, etc.

☺ April 13-17 — 18th IEEE **International Symposium On Real-Time Computing** (ISORC'2015), Auckland, New Zealand. Topics include: object/component/service-oriented real-time distributed computing (ORC) technology; programming and system engineering (ORC paradigms, languages, model-driven development, specification, design, verification, validation, maintenance, time-predictable systems, ...); system software (real-time kernels, middleware support for ORC, extensibility, synchronization, scheduling, fault tolerance, security, ...); applications (embedded systems, real-time object-oriented simulations, ...); system evaluation (timing, dependability, fault detection and recovery time, ...); etc.

April 13-17 — 30th ACM **Symposium on Applied Computing** (SAC'2015), Salamanca, Spain.

- ☺ April 13-17 — **Track on Programming Languages** (PL'2015). Topics include: compiling techniques, domain-specific languages, formal semantics and syntax, garbage collection, language design and implementation, languages for modeling, model-driven development, new programming language ideas and concepts, practical experiences with programming languages, program analysis and verification, programming languages from all paradigms, etc.
- ☺ April 13-17 — **Track on Object-Oriented Programming Languages and Systems** (OOPS'2015). Topics include: aspects and components, code generation and optimization, distribution and concurrency, formal verification, integration with other paradigms, software evolution, language design and implementation, modular and generic programming, secure and dependable software, static analysis, testing and debugging, type systems, etc.
- ☺ April 13-17 — **Track on Software Engineering** (SE'2015). Topics include: software architecture, and software design patterns; maintenance and reverse engineering; quality assurance; verification, validation, testing, and analysis; formal methods and theories; component-based development and reuse; safety, security, and risk management; dependability and reliability; empirical studies, and industrial best practices; applications and tools; etc.

April 13-17 **Track on Programming for Separation of Concerns** (PSC'2015). Topics include: software reuse and evolution of legacy systems; consistency, integrity and security; generative approaches; language support for aspect-oriented and SoC systems; etc.

April 13-17 **Track on Software Verification and Testing** (SVT'2015). Topics include: new results in formal verification and testing, technologies to improve the usability of formal methods in software engineering, applications of mechanical verification to large scale software, etc.

April 20-22 17th **International Real-Time Ada Workshop** (IRTAW'2015), Vermont, New York, USA. In cooperation with AdaCore and Ada-Europe.

April 27-29 7th **NASA Formal Methods Symposium** (NFM'2015), Pasadena, California, USA. Topics include: identifying challenges and providing solutions to achieving assurance in mission- and safety-critical systems, model checking, static analysis, modeling and specification formalisms, model-based development, applications of formal methods to aerospace systems and cyber-physical systems, etc.

April 29-30 10th **International Conference on Evaluation of Novel Approaches to Software Engineering** (ENASE'2015), Barcelona, Spain. Topics include: comparing novel approaches with established traditional practices and evaluating them against software quality criteria, software process improvement, model-driven engineering, application integration technologies, software quality management, software change and configuration management, geographically distributed software development environments, formal methods, component-based software engineering and commercial-off-the-shelf (COTS) systems, software and systems development methodologies, etc.

☺ May 16-24 37th **International Conference on Software Engineering** (ICSE'2015), Firenze, Italy. Topics include: component-based software engineering; debugging, fault localization, and repair; dependability, safety, and reliability; embedded and cyber physical systems; formal methods, verification, and synthesis; middleware, frameworks, and APIs; model-driven engineering; parallel, distributed, and concurrent systems; performance; program analysis; programming, specification, and modeling languages; reverse engineering; security, privacy and trust; software architecture; software economics, management, and metrics; software evolution and maintenance; software modeling and design; software product lines; software reuse; tools and environments; etc.

May 16-24 **Software Engineering Education and Training** (SEET'2015). Topics include: software and system development; new best practices for SEET; innovative curriculum or course formats; blending software engineering and other engineering disciplines, such as electrical engineering and bioengineering; cooperation in education between industry and academia; continuous education to cope with technological change; etc.

May 16-24 **Track on New Ideas and Emerging Results** (NIER'2015). Topics include: startling results that call into question current research directions, bold arguments on current research directions that may be somehow misguided, etc.

May 18 3rd FME **Workshop on Formal Methods in Software Engineering** (FormaliSE'2015). Topics include: integration of FMs in the software development life cycle, ability of formal methods to handle real-world problems, formal methods in a certification context, "lightweight" or usable FMs, application experiences, formal approaches to safety and security related issues, scalability of FM applications, rigorous software engineering approaches and their tool support, case studies developed/analyzed with formal approaches, etc.

May 23 **Workshop on COmplex faUlts and Failures in LargE Software Systems** (COUFLESS'2015). Topics include: applications of software maintenance technologies, reliable software architectures and software engineering techniques, formal aspects (semantics, reasoning, verification), evolution and reverse engineering, refactoring to improve software reliability, software fault location, industrial points of view (experiences, applications, open issues), etc.

May 25-29 29th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2015), Hyderabad, India. Topics include: parallel and distributed algorithms, applications of parallel and distributed computing, parallel and distributed software, including parallel and multicore programming languages and compilers, runtime systems, parallel programming paradigms, programming environments and tools, etc.

☺ May 25 **Workshop on Programming Models, Languages and Compilers for Manycore and Heterogeneous Architectures** (PLC'2015). Topics include: programming models (thread and task based models, data parallel models, stream programming), programming environments for heterogeneous systems, compiler optimizations, runtime systems for multicore processors, application and benchmarks, etc.

May 25 5th NSF/TCPP **Workshop on Parallel and Distributed Computing Education** (EduPar-15). Topics include: novel approaches to incorporating Parallel and Distributed Computing (PDC) topics into undergraduate core courses that are taken by the majority of students in a program; pedagogical tools, programming environments, and languages for PDC; etc.

☺ May 25 20th **International Workshop on High-Level Parallel Programming Models and Supportive Environments** (HIPS'2015). Topics include: the areas of parallel applications, language design, compilers, runtime systems, and programming tools; the areas of emerging programming models for large-scale parallel systems and many-core architectures; new programming languages and constructs for exploiting parallelism and locality; experience with and improvements for existing parallel languages and run-time environments; parallel compilers, programming tools, and environments; programming environments for heterogeneous multicore systems; etc.

June 13-17 ACM SIGPLAN **Conference on Programming Language Design and Implementation** (PLDI'2015), Portland, Oregon, USA. Topics include: programming language research, including the design, implementation, theory, and efficient use of languages; innovative and creative approaches to compile-time and runtime technology, novel language designs and features, and results from implementations; language designs and extensions; static and dynamic analysis of programs; domain-specific languages and tools; type systems and program logics; checking or improving the security or correctness of programs; memory management; parallelism, both implicit and explicit; debugging techniques and tools; etc.

☺ June 18-19 ACM SIGPLAN/SIGBED **Conference on Languages, Compilers, and Tools for Embedded Systems** (LCTES'2015), Portland, Oregon, USA. Topics include: features to exploit multicore, reconfigurable, and other emerging architectures; features for distributed, adaptive, and real-time control embedded systems; language capabilities for specification, composition, and construction of embedded systems; language features and techniques to enhance reliability, verifiability, and security; concurrency; memory management; support for enhanced programmer productivity; support for enhanced debugging, profiling, and exception/interrupt handling; tools for analysis, specification, design, and implementation; 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 as basis for code generation, analysis, and verification; etc.

☺ June 22-23 20th **International Workshop on Formal Methods for Industrial Critical Systems** (FMICS'2015), Oslo, Norway. Topics include: design, specification, code generation and testing based on formal methods; methods, techniques and tools to support automated analysis, certification, debugging, learning, optimization and transformation of complex, distributed, real-time systems and embedded systems; verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability (e.g., scalability and usability issues); tools for the development of formal design descriptions; case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; impact of the adoption of formal methods on the development process and associated costs; application of formal methods in standardization and industrial forums.

♦ June 22-26 20th **International Conference on Reliable Software Technologies - Ada-Europe'2015**, Madrid, Spain. Sponsored by Ada-Europe, in cooperation with ACM SIGAda, SIGBED, SIGPLAN, and the Ada Resource Association (ARA).

☺ June 22 **Workshop on Challenges and New Approaches for Dependable and Cyber-Physical System Engineering** (De-CPS 2015). Topics include: Industrial challenges and experience reports on co-engineering for multiple dependability concerns in CPS engineering; Modeling and analysis of Cyber-Physical Systems (CPS) via contract-based approaches; Tools and methodologies to guarantee safety-related properties, including real-time and mixed-criticality cohabitation; Challenges posed for CPS design

and safety verification by multi-core processors. Deadline for submissions: April 20, 2015.

☺ June 26 **Workshop on Architecture Centric Virtual Integration** (ACVI 2015). Topics include: Architecture Analysis and Design Language; model-driven approaches; distributed, real-time and embedded systems. Deadline for submissions: April 19, 2015.

June 22-26 20th **International Symposium on Formal Methods** (FM'2015), Oslo, Norway. Topics include: interdisciplinary formal methods (techniques, tools and experiences demonstrating formal methods in interdisciplinary frameworks); formal methods in practice (industrial applications of formal methods, experience with introducing formal methods in industry, tool usage reports, etc); tools for formal methods (advances in automated verification and model-checking, integration of tools, environments for formal methods, etc); role of formal methods in software and systems engineering (development processes with formal methods, usage guidelines for formal methods, method integration, qualitative or quantitative improvements); theoretical foundations (all aspects of theory related to specification, verification, refinement, and static and dynamic analysis).

☺ June 22 2nd **International Workshop on Safety and Formal Methods** (SaFoMe'2015). Topics include: formal languages and verification techniques for: design, validation, and verification of safety-critical component-based systems; design and verification of real-time, embedded safety-critical systems; formal methods for safety and security; contract-based design and verification of safety-critical embedded systems; formal methods in the certification of safety-critical systems; formal methods applied in the context of industrial safety-critical case studies; experience reports of using formal methods for certification (e.g., DO 178C); etc.

June 22 4th **International Workshop on Engineering Safety and Security Systems** (ESSS'2015). Theme: "Methods and techniques for constructing large reliable and secure systems". Topics include: methods, techniques and tools for system safety and security; methods, techniques and tools for analysis, certification, and debugging of complex safety and security systems; case studies and experience reports on the use of formal methods for analyzing safety and security systems; etc.

June 23 1st **Formal Methods in Software Engineering Education and Training Workshop** (FMSEET'2015). Topics include: best practices in formal methods education, languages and tools for formal methods education, formal engineering methods versus formal methods, continuous education to cope with technological change, etc.

Jun 29 - Jul 01 12th **International Conference on Mathematics of Program Construction** (MPC'2015), Königswinter, Germany. Topics of interest range from algorithmics to support for program construction in programming languages and systems, such as type systems, program analysis and transformation, programming-language semantics, security, etc.

☺ Jun 29 - Jul 02 14th **International Symposium on Parallel and Distributed Computing** (ISPDC'2015), Limassol, Cyprus. Topics include: multi-cores, methods and tools for parallel and distributed programming, tools and environments for parallel program design/analysis, parallel programming paradigms and APIs, distributed software components, parallel embedded systems programming, scheduling, security and dependability, real-time distributed and parallel systems, etc. Deadline for early registration: May 6, 2015.

Jun 29 - Jul 03 9th ACM **International Conference on Distributed Event-Based Systems** (DEBS'2015), Oslo, Norway. Topics include: software systems, distributed systems, dependability, programming languages, security, software engineering, real-time analytics, embedded systems, enterprise application integration, etc. Deadline for submissions: April 30, 2015 (Doctoral Symposium papers, posters, demos).

July 01-05 39th Annual IEEE **International Computer Software and Applications Conference** (COMPSAC'2015), Taichung, Taiwan. Event includes: symposium on Embedded & Cyber-Physical Environments; symposium on Software Engineering Technologies & Applications; symposium on Security, Privacy and Trust Computing; symposium on Novel Applications and Technology Advances in Computing; symposium on Computer Education and Learning Technologies; etc.

July 06-07 20th Annual **Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2015), Vilnius, Lithuania.

☺ July 06-10 29th **European Conference on Object-Oriented Programming** (ECOOP'2015), Prague, Czech Republic. Topics include: all areas of object technology and related software development technologies, such as concurrent and parallel systems, distributed computing, programming environments, versioning, refactoring, software evolution, language definition and design, language implementation, compiler construction, design methods, design patterns, aspects, components, modularity, type systems, program analysis, specification, verification, security, real-time systems, etc.

☺ July 07-10 27th **Euromicro Conference on Real-Time Systems** (ECRTS 2015), Lund, Sweden. Topics include: Embedded/RT Systems Design; Scheduling Design and Analysis; WCET Analysis, Power, Energy and/or Thermal Aware RTS; RT operating Systems and Middlewares; Network/System-on-Chips; Mixed Criticality Design & Assurance; (Wireless) Sensor networks; RT Applications; Tools and Compilers for embedded systems. Deadline for submissions (workshops): April 24, 2015.

July 07 6th **International Real-Time Scheduling Open Problems Seminar** (RTSOPS 2015). Topics include: Single-, Multi- and Many-core scheduling; New models for real-time systems; Scheduling in cyber-physical systems; Mixed-criticality scheduling; Interactions between WCET (worst-case execution time) analysis and scheduling.

July 07 15th **International Workshop on Worst-Case Execution Time Analysis** (WCET 2015). Topics include: WCET/ETB analysis for multi- and many-core systems; WCET/ETB analysis for COTS processors; Case studies, and industrial experience of WCET/ETB analysis; Timing Analysis and safety standards; Probabilistic timing analysis; Methods and benchmarks for timing analysis evaluation; etc..

July 07 11th **International Workshop on Operating Systems Platforms for Embedded Real-Time Applications** (OSPERT 2015). Topics include: Certification and verification of RTOSs and middleware; Operating system standards (e.g., AUTOSAR, ARINC, POSIX, etc.); Real-time virtualization and hypervisors; Support for (embedded) multiprocessor architectures; Support for component-based development; etc.

July 07 6th **International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems** (WATERS 2015). Topics include: Tools and methods for the analysis of real-time systems; Realistic case studies and reusable data sets; Modelling, analysis and simulation of, possibly mixed-criticality, real-time, distributed, and embedded systems running on multi-core, many-core, massively parallel, or distributed systems; etc.

July 13-16 10th IEEE **International Conference on Global Software Engineering** (ICGSE'2015), Ciudad Real, Spain. Theme: "Solutions for distributed product development and maintenance" Topics include: software design and architecture for distributed development, strategic issues in distributed development, industrial offshoring and outsourcing experiences, tools and infrastructure support for distributed teams, methods and processes for global organizations, etc. Deadline for submissions: May 1, 2015 (industrial abstracts).

July 18-24 27th **International Conference on Computer Aided Verification** (CAV'2015), San Francisco, California, USA. Topics include: theory and practice of computer-aided formal analysis methods for hardware and software systems, algorithms and tools for verifying models and implementations, program analysis and software verification, verification methods for parallel and concurrent hardware/software systems, testing and run-time analysis based on verification technology, applications and case studies in verification, verification in industrial practice, verification techniques for security, etc.

July 20-24 **Software Technologies: Applications and Foundations** (STAF'2015), L'Aquila, Italy. Successor of the TOOLS federated event. Topics include: practical and foundational advances in software technology, from object-oriented design, testing, mathematical approaches to modelling and verification, transformation, model-driven engineering, aspect-oriented techniques, and tools.

July 20-24 9th **International Conference on Tests And Proofs** (TAP'2015). Topics include: the synergy of proofs and tests, to the application of techniques from both sides and their combination for the advancement of software quality; transfer of concepts from testing to proving (e.g., coverage criteria) and from proving to testing; program proving with the aid of testing techniques; verification and testing techniques combining proofs and tests; generation of test data, oracles, or preambles by deductive techniques; automatic

bug finding; case studies combining tests and proofs; formal frameworks; tool descriptions and experience reports; etc.

July 21-23 34th Annual ACM SIGACT-SIGOPS **Symposium on Principles of Distributed Computing** (PODC'2015), Donostia-San Sebastián, Spain.

August 03-05 IEEE **International Conference on Software Quality, Reliability and Security** (QRS'2015), Vancouver, Canada. Merger of SERE conference (International Conference on Software Security and Reliability) and QSIC conference (International Conference on Quality Software). Topics include: reliability, security, availability, and safety of software systems; software testing, verification and validation; software vulnerabilities; formal methods; benchmark, tools, and empirical studies; etc. Deadline for submissions: April 15, 2015 (workshop papers, Student Doctoral Program papers), May 1, 2015 (fast abstracts).

☺ August 20-22 13th IEEE **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2015), Helsinki, Finland. Topics include: parallel and distributed algorithms; tools/environments for parallel/distributed software development; novel parallel programming paradigms; code generation and optimization; compilers for parallel computers; middleware and tools; scheduling and resource management; reliability, fault tolerance, dependability, and security; parallel and distributed systems and architectures; applications of parallel and distributed processing; high-performance scientific and engineering computing; etc.

☺ August 24-28 21st **International European Conference on Parallel Computing** (Euro-Par'2015), Vienna, Austria. Topics include: all aspects of parallel and distributed processing, such as support tools and environments, scheduling, compilers, distributed systems and algorithms, parallel and distributed programming and languages, multicore and manycore programming, theory and algorithms for parallel computation, etc. Deadline for submissions: May 22, 2015 (workshop papers).

August 26-28 41st **Euromicro Conference on Software Engineering and Advanced Applications** (SEAA'2015), Madeira, Portugal. Topics include: information technology for software-intensive systems; model-based development, components and services (MOCS); software process and product improvement (SPPI); embedded software engineering (ESE); cyber-physical systems (CPS); etc.

Aug 31- Sep 04 10th **Joint European Meeting of the Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering** (ESEC/FSE'2015), Bergamo, Italy. Topics include: components and middleware, development environments and tools, distributed software, embedded and real-time software, maintenance and evolution, model-driven software engineering, parallel and concurrent software, reverse- and re-engineering, software architecture, software economics, validation, verification, and testing, etc. Deadline for applications: April 15, 2015 (tutorials), June 1, 2015 (Doctoral Symposium, industrial track), June 7, 2015 (New Ideas track, tool demonstrations, replication packages).

☺ Sep 01-04 **International Conference on Parallel Computing 2015** (ParCo'2015), Edinburgh, Scotland, UK. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments, in particular parallel programming languages, compilers, and environments, tools and techniques for generating reliable and efficient parallel code, testing and debugging techniques and tools, best practices of parallel computing on multicore, manycore, and stream processors, etc.

☺ Sep 01-04 44th **Annual International Conference on Parallel Processing** (ICPP'2015), Beijing, China. Topics include: all aspects of parallel and distributed computing, such as applications, architectures, compilers, programming models, etc.

☺ Sep 01-04 **International Workshop on Embedded Multicore Systems** (EMS'2015). Topics include: programming models for embedded multicore systems; software for multicore, GPU, and embedded architectures; real-time system designs for embedded multicore environments; applications for automobile electronics of multicore designs; compiler for worst-case execution time analysis; formal method for embedded systems; etc. Deadline for submissions: May 1, 2015.

September 01-04 15th **Workshop on Automated Verification of Critical Systems** (AVoCS'2015), Edinburgh, Scotland, UK. Topics include: model checking, specification and refinement, verification of software and hardware, specification and verification of fault tolerance and resilience, real-time systems, dependable systems, verified system development, industrial applications, etc. Deadline for submissions: June 5,

2015 (abstracts), June 12, 2015 (full papers), August 7, 2015 (research ideas). Deadline for early registration: August 18, 2015.

September 06-09 11th **International Conference on Parallel Processing and Applied Mathematics** (PPAM'2015), Krakow, Poland. Topics include: multi-core and many-core parallel computing; parallel/distributed algorithms (numerical and non-numerical); scheduling, mapping, load balancing; parallel/distributed programming; tools and environments for parallel/distributed computing; security and dependability in parallel/distributed environments; applications of parallel/distributed computing; etc. Deadline for submissions: April 26, 2015 (papers).

☺ Sep 06-09 6th **Workshop on Language-Based Parallel Programming Models** (WLPP'2015). Topics include: language and library implementations; proposals for, and evaluation of, language extensions; applications development experiences; comparisons between programming models; compiler implementation and optimization; etc. Deadline for submissions: April 26, 2015.

September 07-08 7th **International Workshop on Software Engineering for Resilient Systems** (SERENE'2015), Paris, France. Topics include: requirements engineering & re-engineering for resilience; frameworks, patterns and software architectures for resilience; design of trustworthy systems; verification, validation and evaluation of resilience; empirical studies in the domain of resilient systems; methodologies adopted in industrial contexts; etc. Deadline for submissions: April 24, 2015.

September 07-11 13th **International Conference on Software Engineering and Formal Methods** (SEFM'2015), York, UK. Topics include: abstraction and refinement; programming languages, program analysis and type theory; formal methods for real-time, hybrid and embedded/cyber-physical systems; formal methods for safety-critical, fault-tolerant and secure systems; software verification and validation; formal aspects of software evolution and maintenance; light-weight and scalable formal methods; tool integration; applications of formal methods, industrial case studies and technology transfer; education and formal methods; etc. Deadline for submissions: May 22, 2015 (workshop papers).

September 13-16 **Federated Conference on Computer Science and Information Systems** (FedCSIS'2015), Lodz, Poland. Deadline for submissions: April 24, 2015 (papers), June 1, 2015 (position papers).

☺ Sep 13-16 5th **Workshop on Advances in Programming Languages** (WAPL'2015). Topics include: compiling techniques; domain-specific languages; generative and generic programming; languages and tools for trustworthy computing; language concepts, design and implementation; model-driven engineering languages and systems; practical experiences with programming languages; program analysis, optimization and verification; programming tools and environments; specification languages; type systems; etc. Deadline for submissions: April 24, 2015 (papers), June 1, 2015 (position papers). Deadline for early registration: July 1, 2015.

Sep 13-16 8th **Workshop on Computer Aspects of Numerical Algorithms** (CANA'2015). Topics include: parallel numerical algorithms; libraries for numerical computations; languages, tools and environments for programming numerical algorithms; paradigms of programming numerical algorithms; etc. Deadline for submissions: April 24, 2015 (papers), June 1, 2015 (position papers).

September 22-25 15th **International Conference on Runtime Verification** (RV'2015), Vienna, Austria. Topics include: monitoring and analysis of software and hardware system executions. Application areas include: safety/mission-critical systems, enterprise and systems software, autonomous and reactive control systems, health management and diagnosis systems, and system security and privacy. Deadline for submissions: April 12, 2015 (abstracts), April 19, 2015 (full papers).

Sep 28 - Oct 01 34th **International Symposium on Reliable Distributed Systems** (SRDS'2015), Montreal, Canada. Topics include: distributed objects and middleware systems, experimental or analytical evaluations of dependable distributed systems, formal methods and foundations for dependable distributed computing, high-assurance and safety-critical distributed system design and evaluation, secure and trusted distributed systems, dependability in cyberphysical systems, etc.

☺October 18-21 24th **International Conference on Parallel Architectures and Compilation Techniques** (PACT'2015), San Francisco, California, USA. Topics include: parallel architectures and computational models; compilers and tools for parallel computer systems; middleware and run time system support for parallel computing; support for correctness in concurrent hardware and software; parallel programming

languages, algorithms and applications; applications and experimental systems studies; etc. Deadline for submissions: August 10, 2015 (ACM Student Research Competition).

October 25-27 ACM SIGPLAN 8th **International Conference on Software Language Engineering** (SLE'2015), Pittsburgh, Pennsylvania, USA. Topics include: techniques for software language reuse, evolution and management of variations (syntactic/semantic) within language families; applications of DSLs for different purposes (incl. modeling, simulating, generation, description, checking); novel applications and/or empirical studies on any aspect of SLE (development, use, deployment, and maintenance of software languages); etc. Deadline for submissions: June 1, 2015 (abstracts), June 15, 2015 (full papers).

☺ October 25-30 ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2015), Pittsburgh, Pennsylvania, USA. Topics include: all aspects of software construction and delivery, at the intersection of programming, languages, and software engineering. Deadline for submissions: June 7, 2015 (Dynamic Languages Symposium), June 30, 2015 (workshops late phase). Deadline for early registration: September 25, 2015.

November 03-06 17th **International Conference on Formal Engineering Methods** (ICFEM'2015), Paris, France. Topics include: abstraction and refinement; program analysis; software verification; software model checking; formal methods for object and component systems, concurrent and real-time systems, cyber-physical systems, for software safety, security, reliability and dependability; tool development, integration and experiments involving verified systems; formal methods used in certifying products under international standards; formal model-based development and code generation; etc. Deadline for submissions: April 19, 2015 (abstracts), April 26, 2015 (full papers).

December 08-11 16th ACM/IFIP/USENIX **International Middleware Conference** (Middleware'2015), Vancouver, Canada. Topics include: design, implementation, deployment, and evaluation of distributed system platforms and architectures for computing, storage, and communication environments; reliability and fault-tolerance; real-time solutions; scalability and performance; programming frameworks, parallel programming, and design methodologies for middleware; methodologies and tools for middleware design, implementation, verification, and evaluation; retrospective reviews of middleware paradigms; etc. Deadline for submissions: May 15, 2015 (abstracts), May 20, 2015 (papers).

December 10 200th birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

The 20th International Conference on Reliable Software Technologies – Ada-Europe 2015 will take place in Madrid, Spain. As per its traditional style, the conference will span a full week, including, from Tuesday to Thursday, three days of scientific, technical and industrial programs, along with tutorials and workshops on Monday and Friday.

The conference will be hosted by ETSIT-UPM, a leading engineering school covering teaching and research in all fields related to Information and Communications Technology, located in Ciudad Universitaria, the main University campus in Madrid near the area of Moncloa.

Overview of the week

Monday	Tuesday	Wednesday	Thursday	Friday
Tutorials & DeCPS Workshop	Keynote talk	Keynote talk	Keynote talk	Tutorials & ACVI Workshop
	Regular session Language technology	Industrial session Critical systems	Regular session Critical systems	
	Vendor session	Regular session Real-time applications	Industrial session Tools at work	
	Industrial session Ada applications	Special session Advances on methods	Regular session Multicore & distributed systems	
	Ada-Europe General Assembly Welcome cocktail	Conference banquet Best paper award Ada Lovelace Celebration	Best presentation award Closing session	

For full details and up-to-date information see the conference web site:

http://www.ada-europe.org/conference2015

Keynote talks

On the three central days of the conference week, a keynote will be delivered as the opening event to address hot topics of relevance in the conference scope. The keynote speakers include Jon Pérez, head of embedded systems research at IKERLAN, who will present his work on *EC-61508 Certification of Mixed-Criticality Systems Based on Multicore and Partitioning*; Javier Rodríguez, from Siemens Rail Automation, who will talk on his experience in *Software Development of Safety-Critical Railway Systems*; and Andras Balasz, to talk on *The Central On-board Computer of the Philae Lander in the Context of the Rosetta Space Mission*, based on his long experience in the Rosetta-Philae and other space projects.

Exhibition

From Tuesday until Thursday the coffee break area will feature exhibitor booths, project posters, reserved vendor tables, and general networking options. If interested, contact the exhibition chair (see website).

Tutorials

Monday 22 June

Morning	T1 - *J.P. Rosen* Access types and Memory Management in Ada 2012	T3 - *B. Moore, S.Michell* Parallelism in Ada, Today and Tomorrow	T4 - *F. Cazorla, T. Vardanega, J. Abella, M. Pearce* Probabilistic Timing Analysis
Afternoon	T2 - *J.P. Rosen* Designing and Checking Coding Standards for Ada		

Friday 26 June

Morning	T5 - *J. Sparre-Andersen* Ada 2012 (Sub)Types and Subprogram Contracts in Practice	T7 - *W. Bail* Software Measures for Dependable Software Systems	T9 - *P. Rogers* Real-Time/Embedded Programming with Ada 2012
Afternoon	T6- *E. Briot, B. Brosgol* When Ada Meets Python: Extensibility through Scripting	T8- *W. Bail* Software Design Concepts and Pitfalls	

Workshops

Two major workshops are taking place in connection with the conference.

- Monday 22 June: *Challenges and new Approaches for Dependable and Cyber-Physical Systems Engineering*.
- Friday 26 June: *Architecture Centric Virtual Integration*.

Social program

A welcome cocktail will be served at the conference location on Tuesday evening, followed by a bus transport to central Madrid. The participants will then be let free to have dinner in the city, where there are plenty of tapas bars and traditional restaurants within reach.

The conference banquet will take place on Wednesday evening at Club de Campo Villa de Madrid, a country club located at the outskirts of the city, with magnificent views. Superb dinner will be served, and we will give the award for best paper as usual.

Further information

The conference web site gives full and up to date details of the program and the venue, including travel advice, maps and hotels close by. A limited number of rooms have been blocked for the conference in the Exe Moncloa hotel in Moncloa, but please be prompt in booking your accommodation since the demand is high in Madrid for the conference dates. Online registration is open, with reduced fees until June 7.

Ada 2015 sponsors

(preliminary list)

AdaCore

Ellidiss Software

RAPITA SYSTEMS LTD

Ada europe

SpAda in

acm In-Cooperation

POLITÉCNICA

ETSIT UPM

Ada-Europe 2015

An Invitation to Join Ada-Europe

What is Ada-Europe?

Ada-Europe is an international organization, set up to promote the use of Ada. It aims to spread the use and the knowledge of Ada and to promote its introduction into academic and research establishments. Above all, Ada-Europe intends to represent European interests in Ada and Ada-related matters.

In its current form, Ada-Europe was established in 1988. As there is no European legal framework to govern such organizations, it was established according to Belgian Law. Currently, the member organizations are: Ada-Belgium, Ada in Denmark, Ada-Deutschland, Ada-France, Ada-Spain, Ada in Sweden and Ada-Switzerland. Individual members of these organizations can become indirect members of Ada-Europe. Direct membership is available to individuals in countries without national member organization.

What does Ada-Europe do?

The best-known of Ada-Europe's activities is its annual conference. These conferences usually attract around 100 participants. They involve three days of lectures and presentations, and provide the perfect opportunity to discuss new information and exchange experiences with fellow Ada users. As well as the usual conference features, you have the opportunity to attend an additional two days of tutorials dealing with specialist Ada matters. The conference also hosts an exhibition, where Ada-related products are presented.

Ada-Europe offers a framework for setting up working groups and task groups to discuss and investigate technical aspects of using Ada on a European basis. It provides grants for Ada-related conferences and activities.

The members of Ada-Europe also receive the quarterly Ada User Journal, produced by Ada-Europe. This journal contains Ada-related papers, experience reports, details of past, present and future Ada events and activities, and reviews of new publications and products. The journal is usually distributed via the national member organizations, but can also be mailed directly at additional postage costs.

A reduced registration fee at the annual Ada-Europe conference is an additional benefit to direct and indirect members registered with Ada-Europe by their national organizations. On a semi-regular basis, Ada-Europe "surprises" its individual members with useful material: for example, in 2006 the offer of the Ada 2005 Reference Manual, or more recently, the discounted price for the Ada 2012 Reference Manual and for the Programming in Ada 2012 book.

How to become a member of Ada-Europe?

Individuals

If you want to become a member of Ada-Europe, please join your national Ada organisation and become an indirect member of Ada-Europe. In some countries, indirect membership in Ada-Europe is automatically part of your national membership; in other countries, it is an optional element of your national membership.

As benefits you will:

- receive a free copy of the quarterly Ada User Journal, distributed via the national Ada organisations
- have a reduced registration fee at the annual Ada-Europe conference (exceeding the cost of your indirect membership)
- access free or discounted books and other resources
- participate in both technical inititativaes as well as community building actions.

Your benefits run from April to March of the following year.

If your country does not have a national Ada organisation, you can contact the Secretary of Ada-Europe to become a direct member of Ada-Europe. Your benefits are the same as for indirect members, except that the Journal is shipped directly to you.

Institutions

National Ada organisations are the primary promoters of corporate memberships. In case a national Ada organisation exists in your country, it can offer its corporate members to designate individuals as indirect members of Ada-Europe at the Ada-Europe individual indirect membership fee (plus any fees that your national organization charges).

In case no national organisation exists in your country, corporate membership may be established directly with Ada-Europe.

Further information

For further information please refer to Ada-Europe's website at http://www.ada-europe.org, or contact the General Secretary of Ada-Europe.

National organizations contacts are available on the last page.

Ada Lovelace: Victorian Computing Visionary

Suw Charman-Anderson

Abstract

This year is the 200th anniversary of Ada Lovelace's birth, a woman whose achievements lay unrecognised for more than a hundred years, and who even now is not the household name she ought to be.

Back in 2009, I was searching for a name for a day of blogging about women in tech that I was planning, and a friend suggested that Ada Lovelace would be the perfect candidate. I had never heard of her, but the more that I read about her, the more I realised that she would was the perfect tech heroine: Not only did she write the first computer program, it was for a mechanical computing machine - Charles Babbage's Analytical Engine - which was not and never would be built.

But Ada did more than write a program to calculate Bernoulli numbers, she also realised that the Analytical Engine was much more than just a glorified calculator. She saw that it could do very human things, like make music or graphics, given the right inputs and algorithms. She foresaw what we now call computer science but, like many women in that field, her legacy is questioned, her capabilities disparaged, and her accomplishments belittled. There could not be a better figurehead for modern women in technology than Ada Lovelace.

1 Introduction

The idea that the 1840s saw the birth of computer science as we know it today may seem like a preposterous one, but long before the Bombe, the Colossus or the Harvard Mark I, long before any computer was actually built, came a remarkable woman whose understanding of computing remained unparalleled and unappreciated for 100 years. Brought up in an era when women were routinely denied education, she saw further into the future than any of her male counterparts, and her work influenced the thinking of one of World War II's greatest minds.

Born The Honourable Augusta Ada Byron, the woman we know of today as Ada Lovelace began her life in a turbulent household. She was the only legitimate daughter of George Gordon Byron, 6th Baron Byron and Anne Isabella Milbanke, 11th Baroness Wentworth, or Annabella as she called herself. Their short marriage imploded just a month after Ada was born.

Annabella was an incredibly intelligent woman who was educated by former Cambridge University professors in classical literature, philosophy, science and maths. She particularly enjoyed maths and Byron called her his "princess of parallelograms". But Annabella was also a stiff, religious woman with strict morals, and was sometimes described as cold and prim.

Byron, on the other hand, was the original cad. He was "mad, bad and dangerous to know" according to Lady Caroline Lamb, Annabella's cousin, who had an affair with Byron before his marriage. Annabella probably should have taken this as the warning it was — Lamb never really got over her break-up with Byron — but perhaps she instead took it as a challenge. Maybe she saw Byron as a soul that needed saving from his lascivious and immoral ways. Whatever her motivations, Annabella and Byron wed in January 1815 and Ada was born on December 10 that same year.

The marriage, however, wasn't a happy one. Byron was moody, drank too much and behaved erratically, having at least one affair with a London chorus girl called Susan Boyce. There were rumours of violence and abuse. And then the financial troubles hit. Byron suggested that Annabella remove herself to her parents' estate at Kirkby Mallory, and take Ada with her, whilst he sorted things out.

Worried that he had succumbed to madness, Annabella engaged a physician to visit the family and secretly assess Byron's state of mind. The physician recommended that she do as Byron wished, and in January 1816 the couple separated. Although their separation began amicably enough, it soon turned acrimonious and Byron left England for Italy to escape a burgeoning sexual scandal. Ada never met her father, and he died when she was eight years old.

2 Young Ada

Parenting styles were different in the early 19th century, and Annabella wasn't the doting mother that we might these days assume she should be. Indeed, Ada was brought up mostly by her grandmother, Judith, The Honourable Lady Milbanke. Annabella didn't seem to show much affection for her daughter, referring to Ada as 'it' in one letter to her mother:

"I talk to it for your satisfaction, not my own, and shall be very glad when you have it under your own."

Judith died when Ada was six, and the young girl was then looked after by a series of nannies, a common practice at the time, and educated by tutors that her mother appointed.

Ada loved machines. She spent hours poring over diagrams of new inventions and eagerly devouring any new periodical journals she could get her hands on. She began to design boats and steam-powered flying machines for her own amusement.

This unusual preoccupation was encouraged by Annabella, who ensured that Ada was taught by some of the very finest minds in England. Having enjoyed a first class education

herself, Annabella was determined that Ada should have the same, arranging for a series of teachers to give Ada a solid grounding in science and mathematics.

Her motivation wasn't entirely focused on expanding Ada's mind, however: Annabella was terrified that Ada might have inherited the madness of her father. She saw the close study of mathematics and science as a way to instil some mental discipline and, hopefully, drive out any demons that might otherwise plague Ada.

Indeed, in later life, Ada herself said that her study of mathematics helped with the mental instabilities that she does indeed seem to have inherited from her father. She wrote to her husband that "nothing but a very close and intense application to subjects of a scientific nature now seems at all to keep my imagination from running wild, or to stop the void which seems to be left in my mind." However, Ada also wrote to her tutor De Morgan's wife that she had determined that "too much mathematics" had caused her to have a breakdown, so her internal jury was obviously out on maths' effectiveness for the control of her mental problems.

That tutor, Augustus De Morgan, was one of Ada's most important teachers. He was a mathematician at the forefront of the emerging field of symbolic logic. It was, without doubt, De Morgan who encouraged Ada to further study mathematics, and she impressed him mightily with her capabilities. Had Ada been a man, he said, she would have had the potential to become "an original mathematical investigator, perhaps of first-rate eminence".

But De Morgan worried that her focus on maths was damaging her health. Ada had been a sickly child, suffering headaches that affected her vision from around age eight. Then in 1829, when she was 13, she caught the measles which left her paralysed and confined to bed for a year. She did recover, but it was a slow, arduous journey to the point where, in 1831, she could walk again, on crutches.

De Morgan worried that her health would suffer further if she studied too hard. He said of her maths problems that, "the very great tension of mind which they require is beyond the strength of a woman's physical power of application."

But Ada did apply herself and she did conquer her maths problems.

Another of Ada's tutors was Mary Somerville, the Scottish astronomer and mathematician. Mary had become famous in 1831 when she published The Mechanism of the Heavens, a translation of the five volume Mécanique Céleste by Pierre-Simon Laplace. Her translation was published by the wonderfully named Society for the Diffusion of Useful Knowledge and soon Somerville was a household name, yet she was a modest woman who said only that she had "translated Laplace's work from algebra into common language".

In 1833 Mary introduced Ada to another mathematician, Charles Babbage. Ada was 18 and Babbage was 42. It was a friendship that would change Ada's life.

3 Big tables of numbers

Charles Babbage was an inventor and mechanical engineer; given Ada's fascination for machines, it was only natural that the two become fast friends. Ada was captivated by Babbage's inventions and he was impressed by her intellect, analytical skills and mathematical ability.

Babbage was working on a mechanical adding machine that he called the Difference Engine. At the time, any maths that required logarithmic or trigonometric functions forced the mathematician to refer to large tables of numbers that had been worked out by hand. Unfortunately, these tables were prone to error and if an incorrect value was used in the calculations the mathematician's result would also be incorrect. It was Babbage's mission to use the Difference Engine to calculate these tables flawlessly.

The British Government was most interested in such a machine — or rather they were most interested in error-free log and trig tables that could be relied upon to give the correct answer. They invested some £17,000 — now equivalent to about £1.7 million — in the Difference Engine, hoping that Babbage would build it and start producing these vital tables.

Babbage, however, had other ideas. He gave up on the Difference Engine before it was finished and started work on a more complex machine, the Analytical Engine. The British Government was unimpressed and refused to fund Babbage's new project, much to his disgust. It seems Babbage never quite grasped the idea that his funding was dependent on the production of those perfect tables of numbers, tables which never came to exist. Had he delivered, perhaps he would have found it easier to continue raising money.

The Analytical Engine was, however, a major leap forward, so it's easy to understand why Babbage might have abandoned the Difference Engine in the light of this new, more powerful machine. It was a general purpose computing machine that had all the elements of a modern computer, including an arithmetical unit, conditional branching and loops, and integrated memory.

It could also be programmed to do complex computations using punched cards, just like the Jacquard loom and the early modern computers built in the 1940s such as the Harvard Mark I. Babbage even designed a printer to go with it.

But, like its predecessor, the Analytical Engine was never built. In fact, Babbage never quite finished the design, tinkering with it throughout his life.

4 Marriage and family life

On 8 July 1835, aged just 19, Ada became Baroness King when she married William King, the 8th Baron King. King was ten years her senior, and over the next four years, the couple had three children: Byron, born May 1836; Anne Isabella, born September 1837; and Ralph Gordon, born July 1839. After the birth of Anne Isabelle, also called

Annabella like her grandmother, Ada fell ill again for several months.

In 1838, King was created 1st Earl of Lovelace, and Ada became the Right Honourable the Countess of Lovelace. In correspondence she signed herself Augusta Ada Lovelace, or AAL, and we know her today simply as Ada Lovelace.

Her marriage to King was, in some ways, a mirror image of her parents' marriage, but with their roles reversed. King was a bit humourless, possibly even abusive and was described at the time as a "figure more of fear than affection". Lovelace was an unconventional woman, fiercely intelligent and independent. She became a materialist and, in her later years, an atheist, which was quite in opposition to the strict Christianity of her mother and husband.

She also found it very easy to strike up friendships and often found herself the object of men's affections. When she was a teenager, she had had an affair with a tutor, with whom she had attempted to elope. But his relatives had recognised her — she was a well-known society figure because of her father and station — and her mother had covered up the scandal.

Later in life her children's tutor, William Benjamin Carpenter, attempted to coax her into another affair. Once she realised what was going on, she ended her association with him. However, her easy-going, charming nature and willingness to converse and correspond with members of the opposite sex meant that there were often rumours of affairs in amongst the court gossip. The fact that she was Byron's daughter cannot have helped matters either.

5 The Menabrea paper

Despite any differences in personality and outlook, King did support Ada's work and ambitions, much more so than many men would have at the time. Throughout this period Lovelace and Babbage's friendship flourished and she studied his plans for the Analytical Engine in depth, becoming an expert on its workings.

In 1842, Babbage gave a lecture about the Analytical Engine at the University of Turin. In the audience was an Italian engineer, Luigi Menabrea whose notes were eventually published in the Bibliothèque Universelle de Genève. Babbage's friend Charles Wheatstone then commissioned Lovelace to translate the paper, which had been written in French, which she did. Babbage then asked Lovelace to expand on the original, "as she understood the machine so well". Lovelace set to work, adding individual notes, each labelled with a letter. When she was done, she had tripled the original paper's length.

In her notes, Lovelace suggests that symbolic logic, which she'd learnt about from De Morgan, could be applied to the Analytical Engine. And in Note G, the final note, she described a number of what we would now call programs to enable the Analytical Engine to do computations without the answers having been calculated by human hand first.

At the time, machines such as the automata which mimicked humans and animals using clockwork were well known at court. In the middle of the 18th Century, Frenchman Pierre Jaquet-Droz had masterminded the creation of three automata: the musician, the draughtsman and the writer. All three, which still exist and still work, can carry out the tasks for which they are named, but in a limited way. A complex series of cams — shaped, rotating disks — control the automata, allowing them to act out a pre-determined series of movements which results in music, drawings and writing. Lovelace would have at the very least seen the Silver Lady, a female automaton which could "bow and put up her eyeglass at intervals, as if to passing acquaintances" which Babbage had bought.

This is conjecture, but perhaps Lovelace was keen to stress that the Analytical Engine produced results without human interference not just to draw a distinction between it and the earlier Difference Engine, but also automata. The Difference Engine had never been built, but automata were a relatively common court spectacle and might have been the first point of comparison for people unfamiliar with Babbage's work. Whatever her motivation, Lovelace was right: The Analytical Engine really was in a league of its own.

"The bounds of arithmetic were however outstepped the moment the idea of applying the cards had occurred;" she wrote, "and the Analytical Engine does not occupy common ground with mere 'calculating machines.' It holds a position wholly its own; and the considerations it suggests are most interesting in their nature. In enabling mechanism to combine together general symbols in successions of unlimited variety and extent, a uniting link is established between the operations of matter and the abstract mental processes of the most abstract branch of mathematical science."

One set of numbers that Lovelace focused on were Bernoulli Numbers, as suggested to her by Babbage. These are a complex numerical system first described by the Swiss mathematician Jakob Bernoulli, who died in 1705. But really, she could have chosen any complex series — the point of the exercise was not to find out what the Bernoulli Numbers were, but to show that they could be calculated by the machine, on its own, from first principles.

Note G described how to break down the algebra into simple formulae, and then how to code those formulae as instructions for the Analytical Engine. Although there were earlier sketches for programs that had been prepared by Babbage, Lovelace's were the most elaborate and complete, and they were the first to be published.

It is for this achievement that Lovelace is known as the first computer programmer: She was the first person to write and publish a full set of instructions that a computing device could use to reach an end result that had not been calculated in advance.

6 A bigger vision

Lovelace's understanding of Babbage's Analytical Engine was so deep that it surpassed that of Babbage himself. She looked beyond those huge tables of perfect numbers that Babbage wanted the machine to calculate and saw its full potential.

It could, Lovelace suggested, create music or graphics, were it to be given the right inputs. The Analytical Engine, she wrote, "weaves algebraic patterns just as the Jacquard loom weaves flowers and leaves."

This idea of a general computer, more than Note G, was a groundbreaking one. It is clear from her notes that this was not just a random flight of fancy, it was a concept she had thought hard about and she had a solid grasp on the theory behind it. She wrote:

"[The Analytical Engine] might act upon other things besides *number*, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine. Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."

Of course, neither Lovelace nor Babbage had the benefit of our modern terminology, and Lovelace had to define her concepts carefully so that her intent remained clear. For example:

"It may be desirable to explain, that by the word operation, we mean any process which alters the mutual relation of two or more things, be this relation of what kind it may. This is the most general definition, and would include all subjects in the universe."

The concept of a general computer that could do anything, given the right program and inputs, was an extraordinary leap for Lovelace to make and one that many of her male peers struggled to understand. It is no exaggeration to say that she was 100 years ahead of her time.

7 Faraday

Even after Lovelace had made this huge conceptual leap, essentially describing much of what we consider fundamental to modern computer science, she continued to expand on the education her mother had arranged for her. Her personality and desire for knowledge is nowhere epitomised more than in a letter to Michael Faraday that she wrote in 1844, aged 28.

Faraday, one of the most influential scientists in history especially in the field of electricity, was a devout Christian and a Sandemanian, a denomination of the Church of Scotland. He was a humble but self-disciplined man, as well as an eloquent and passionate public speaker.

Ada was keen to become his pupil. In fact, 'keen' might have been an understatement. She was, in modern terms, a bit of a fangirl, saying in a letter to him that she "entertain[ed] an esteem little short of reverence" for him. In another letter, dated 1844 — thus making her 28 at the time and him 53 — she wrote (emphasis as per original):

Dear Mr Faraday,

*I am exceedingly **tickled** with your comparison of yourself to a **tortoise**. It has excited all my **fun** (& I assure you I have **no little** of **that** in me).*

*I am also struck with the **forcible truth** of your designation of **my character** of mind:*

*"**elasticity of intellect**".*

*It is indeed the **very truth**, most happily put into language.*

You have excited in my mind a ridiculous, but not ungraceful, allegorical picture, viz:

*that of a quiet demure plodding **tortoise**, with a beautiful **fairy** gambolling round it in a thousand radiant & varying hues; the tortoise crying out, "Fairy, fairy, I am not like you. I cannot at pleasure assume a thousand aerial shapes & expand myself over the face of the universe. Fairy, fairy, have mercy on me, & remember I am **but a tortoise**".*

Babbage, for his part, tried to put in a good word for Lovelace with Faraday, calling her the "Enchanted Maths Fairy", and writing:

My dear Faraday,

I am not quite sure whether I thanked you for a kind note imputing to me unmeritedly the merit of a present you received I conjecture from Lady Lovelace.

I now send you what out to have accompanied that Translation.

So you will now have to write another note so that Enchantress who has thrown her magical spell around the most abstract of Sciences and has grasped it with a force which few masculine intellects (in our own country at least) could have exerted over it. I remember well your first interview with the youthfull fairy which she herself has not forgotten and I am gratefull to you both for making my drawings rooms the Chateau D'Eu of Science.

No amount of fairies or enchantresses could change Faraday's mind, however, and he never did acquiesce to her request to tutor her.

8 Early to rise, early to bed

Lovelace's brilliance had become obvious very early on in her life but, however strong the powers of her mind, she couldn't prevent her body's frailty from betraying her. She had suffered and recovered from cholera, but now she developed uterine cancer, an illness from which she would never recover.

Annabella sat constantly beside Ada's bedside as her condition deteriorated, and — perhaps out of concern for her daughter's immortal soul — did all that she could to

force her to convert to Christianity. She even went so far as to withhold her morphine, the painkiller that would have made Ada's suffering a little more bearable. It is said that Ada did indeed convert, but how much stock can be put in a deathbed conversion under the duress of an agonising death we cannot say.

Ada died on 27 November 1852 at just 36 years old, the same age as her father.

Although Annabella had forbidden Ada from seeing a portrait of Byron until she was 20, Ada had come to know him, as much as she could given that she had been so young when he died. She had read his poetry, though she cared little for it, saying "I shall be a better analyst [mathematician] than my father ever was a poet!"

The older Ada got, the more she identified with her father, writing once that she understood his impulses as she too hated any kind of restraint. In the end, she chose to be buried next to him at the Church of St. Mary Magdalene in Hucknall, Nottingham.

9 A computing legacy

It's fair to say that Babbage's Analytical Engine was a computing evolutionary cul-de-sac. It was never built. Lovelace never had the opportunity to test her program on it. And Babbage never produced those large, error-free tables of numbers that the British Government so desired.

But Lovelace's ideas found their way into modern computing via Alan Turing. During WWII, as he was working at Bletchley Park on decoding German communications, Turing discovered Lovelace's Menabrea translation and its attendant notes. They were critical documents that helped to shape his thinking.

Indeed, in his seminal paper Computing Machinery and Intelligence Turing explored the question "Can machines think?", promptly launching the field of artificial intelligence. He also listed "contrary views" on his position that machines could at least imitate thinking, and discusses what he calls Lady Lovelace's Objection.

Lovelace had written, "The Analytical Engine has no pretensions to originate anything. It can do whatever we know how to order it to perform", which might be taken to mean that her position was that machines could not learn, or create anything original. However, Turing points out that "the evidence available to Lady Lovelace did not encourage her to believe" that machines could be so capable.

He goes on to say, "The Analytical Engine was a universal digital computer, so that, if its storage capacity and speed were adequate, it could by suitable programming be made to mimic the machine in question. Probably this argument did not occur to the Countess or to Babbage. In any case there was no obligation on them to claim all that could be claimed."

Turing, of course, was working a century after Lovelace and could benefit not just from all of the technological developments and advances in scientific knowledge that had occurred in that time, but also from the different culture that he inhabited.

Lovelace's culture, remember, still hadn't developed a concept of machine much beyond the automaton, the clockwork ensemble that mimicked life, but could never create new behaviour. The Writer could write only what it was told to write. The Analytical Engine, on the other hand, could produce an answer that it had worked out for itself based on its inputs and programming.

It is reasonable to argue, however, that it was not just that Lovelace had seen no evidence that machines could act as originators, but that those machines which had appeared to act thus had been frauds. One automaton, in particular, had appeared capable of replicating human thought: The Mechanical Turk was a machine that could not only play chess, but could also reliably win against grandmasters. It toured Europe from 1770 until 1854 and won nearly every match it played. The Mechanical Turk even won a match with Babbage, who said he was sure it was a trick, but he couldn't see how.

About a decade or so before Lovelace published her translation and notes on the Analytical Engine, the Turk was exposed as a hoax. It was not, in fact, an automaton at all, but a machine driven by a human sitting, probably very uncomfortably, in the box at its base. One can't help but wonder if Lovelace's assertion that the Analytical Engine could not originate was as much based on a desire to differentiate it from automata, real or hoaxed, as a belief that it was not possible.

10 Modern objections

Like many woman who have contributed greatly to the fields of science, technology, engineering and maths through the centuries, Lovelace's achievements have often been either downplayed or rejected by modern voices. There are two main objections. Firstly, it is said that Lovelace didn't understand calculus and thus, the logic goes, could not have had the capacity to prepare the Bernoulli program.

It is true that Lovelace struggled with calculus. She wrote to De Morgan in some frustration about the chapter on "notation of functions" that she was studying.

"I do not know when I have been so tantalised by anything," she said, "and should be ashamed to say how much time I have spent up on it, in vain. These functional equations are complete will-o'-the-wisps to me. The moment I fancy I have really at last got hold of something tangible and substantial, it all recedes further and further & vanishes again into thin air."

But calculus caused a lot of trouble for even seasoned mathematicians. Charles Dodgson, better known as the author Lewis Carroll, studied maths for four years at Oxford University, came top of his class and went on to lecture maths there. Dodgson said of the subject, "Talked over the Calculus of Variations with Price today; I see no prospect of understanding the subject at all."

It feels a little harsh to criticise Lovelace for finding calculus tricky when Dodgson, who had the benefit of a formal education at one of the best universities in the world, also found it problematic. Indeed, biographer Dr Betty Alexandra Toole, author of Ada: the Enchantress of Numbers, told me that she showed the De Morgan correspondence to the late Dr Steven Deliberto, then Vice Chairman of the Mathematics department at Berkeley, who stated that Ada was studying what was then considered the forefront of calculus. The fact that we teach calculus at school now should not influence our assessment of the position and understanding of calculus in the 19th century.

A second, and more damning, objection to calling Lovelace the First Computer Programmer comes from the idea that she did not actually write the Bernoulli program. This is a theory that has been put forward by historians, and even by some of her biographers.

Historian Bruce Collier wrote in his 1990 book, The Little Engine That Could've:

It would be only a slight exaggeration to say that Babbage wrote the Notes to Menabrea's paper, but for reasons of his own encouraged the illusion in the minds of Ada and the public that they were authored by her. It is no exaggeration to say that she was a manic depressive with the most amazing delusions about her own talents, and a rather shallow understanding of both Charles Babbage and the Analytical Engine… To me, [correspondence between Ada and Babbage] seems to make obvious once again that Ada was as mad as a hatter, and contributed little more to the Notes than trouble.

Collier isn't alone in his assertion. Allan G. Bromley and Doron Swade both claimed that Babbage did the work in the years before the 1842 publication of Lovelace's translation. Benjamin Woolley says that Lovelace made just "some contribution".

It may be that the confusion, which we'll generously call it, comes from Babbage's own autobiography which he wrote when he was nearly 80. In it, he said (emphasis mine):

The elementary principles on which the Analytical Engine rests were thus in the first instance brought before the public by General Menabrea.

Some time after the appearance of his memoir on the subject in the "Bibliothèque Universelle de Genève," the late Countess of Lovelace informed me that she had translated the memoir of Menabrea. I asked why she had not herself written an original paper on a subject with which she was so intimately acquainted? To this Lady Lovelace replied that the thought had not occurred to her. I then suggested that she should add some notes to Menabrea's memoir; an idea which was immediately adopted.

We discussed together the various illustrations that might be introduced: I suggested several, but the selection was entirely her own. So also was the algebraic working out of the different problems, except, indeed, that relating to the numbers of Bernoulli, which I had offered to do to save Lady Lovelace the trouble. This she sent back to me for an amendment, having detected a grave mistake which I had made in the process.

The notes of the Countess of Lovelace extend to about three times the length of the original memoir. Their author has entered fully into almost all the very difficult and abstract questions connected with the subject.

We have to ask what Babbage meant by "algebraic working out". The Bernoulli note is made of up equations, and a table and diagram which describes how the punch cards should be prepared for the programming of the Engine. It is the table and diagram that are the program, not the equations. So even though Babbage worked on the equations — and he did so to save Lovelace time, not because she couldn't do them herself — that doesn't mean Lovelace didn't write what we now consider to be the program.

Their correspondence illuminates the matter further. Lovelace had herself written, "I want to put in something about Bernoulli's Number, in one of my Notes, as an example of how an explicit function may be worked out by the engine, without having been worked out by human head and hands first."

She wrote Note G and sent it to Babbage for his feedback. Babbage, sadly, lost it and had to ask Lovelace to have another go, to which she replied, "I suppose I must set to work to write something better, if I can, as a substitute, the same precisely I could not recall."

Babbage responded to this new version, "I like very much the improved form of the Bernoulli Note but can judge of it better when I have the Diagram and Notation."

It would have been a most peculiar exchange were the assertion that Babbage wrote the program to be true. Who would say that they would be able to judge something better once they'd been given more information by someone else if they had written it themselves?

A more realistic interpretation is that Babbage and Lovelace collaborated closely, discussing and refining their ideas, Babbage working on some parts, Lovelace on others. That does not detract from her achievements, nor does it lend weight to the claim that Babbage alone wrote the Bernoulli program.

11 The perfect figurehead for women in STEM

In 2009, when Ada Lovelace was first suggested to me as a figurehead for a day celebrating the achievements of women in technology, she seemed like a great choice. Here was the first ever computer programmer. Not the first woman, the first person. How perfect!

It was only as I discovered more about her story and especially the way in which many modern voices, even including some of her biographers, have downplayed or denied her achievements that I realised what an appropriate choice I had made. Although much has changed in the last

two hundred years, many women still find that their contributions to our understanding of the world are either ignored or the accolades go to their male colleagues.

Were Ada alive today, I think she would recognise the problems faced by her female peers. But she'd also recognise our modern computing machines as the very embodiment of her ideas, and she'd immediately set about learning how they worked and how to program them. She didn't let the conventions of her day slow her down, and she certainly wouldn't let modern prejudices get in the way either.

This is an extract from the ebook A Passion for Science: Stories of Discovery and Invention.

About the Author

Suw Charman-Anderson is the founder of Ada Lovelace Day, an international celebration of the achievements of women in science, technology, engineering and maths. Each year, ALD hosts flagship science cabaret event in London, whilst around the world independent groups put on their own events. Last year, Ada Lovelace Day Live! itself was hosted by the venerable Royal Institution, and there were over 65 grassroots events in 13 different countries on five continents.

Suw is a social technologist and, as one of the UK's social media pioneers, has worked with clients worldwide. A freelance journalist, she has written about social media, technology and publishing for The Guardian, CIO Magazine and Forbes. She also co-founded the Open Rights Group in 2005.

Learn more about Ada Lovelace Day at www.findingada.com.

Persistent Containers with Ada 2012

Jacob Sparre Andersen
Jacob Sparre Andersen Research & Innovation, Jægerparken 5, 2. th., 2970 Hørsholm, Danmark

Abstract

Persistent objects form a general and very useful method for storing internal program data between executions of a program. And as [1] points out, Ada is an excellent language for implementing persistent objects. This paper introduces Ada 2012 [2] style containers with a memory mapped file as the backing. The persistent containers allow an application programmer to make objects stored in a container persistent with only small modifications to the source text of the application. The performance and reliability of the implementation is compared with serialisation and with persistent storage pools [3].

1 Introduction

This paper is the result of a review of my paper on persistent storage pools [3] with two things in mind: Benefitting from new features of Ada 2012, and avoiding the conflict with address space layout randomisation inherent in [3]. There are two basic ideas behind this paper. The one is that memory-mapping is an extremely efficient I/O method. The other is that Ada 2012 style containers is a much more programmer-friendly way of storing objects than explicitly allocating them on a storage pool.

These two ideas combined allow us to create a container, which allocates space for its contents in a part of virtual memory which is mapped to a file, and thus automatically stored. Using a persistent container or one of the containers declared under `Ada.Containers` only differs in the call to bind the container to its backing file, making this technique very easy to use.

In section 2 the interface to and use of persistent containers is presented. Section 3 describes the actual implementation. In section 4 a comparison to other techniques for implementing persistence, including an experimental comparison with serialisation, is presented. Finally, in sections 5 and 6, we summarise, conclude and indicate how persistent containers can be enhanced compared with this technique covered in this paper. The full source code of the system, as well as demonstration programs are available from `http://www.jacob-sparre.dk/persistent-containers`.

2 An interface for persistent objects

The concept of persistent objects is about maintaining a collection of objects created by an application from one execution of the application to the next. Two of the techniques devised for this purpose are serialisation, where the objects are written to a file represented as a stream, and storage in a database, where the objects in the process memory are simply buffers for data stored in a relational database.

The reader is pointed to [4] for a general and thorough presentation of persistent objects in relation to Ada. The ideal for persistence implementations is generally that it is "transparent" (or "orthogonal"). I.e. the programming environment/compiler does all the work of saving and loading the full program state between executions.

The interface presented here is not *quite* that easy to use, but it has the benefit of allowing the programmer to control which parts of the program state is persistent; objects stored in **persistent containers** are persistent, while all other objects are not. The persistent containers have to be bound to a file, but once that is done, the container library and the operating system takes care of maintaining a persistent copy of the objects stored in the persistent containers. The Ada 2012 aspect "`Implicit_Dereference`" allows containers to give users safe, direct access to objects stored in them (without having to copy the objects out of the containers first), thus letting the program work directly on the persistent version of an object.

There are no special requirements on the types of objects made persistent, except that system addresses, access types and objects related to tasking and synchronisation should not be stored in persistent containers.

At the time of writing, the library of persistent containers is limited to a linked list container proof-of-concept.

2.1 Package specification

The generic package "Persistent_Containers.Linked_List" declares a persistent linked list container. Comparing it with "Ada.Containers.Doubly_Linked_List", the main difference[1] is the procedure used to bind a list to the file used to persist the container contents:

```
procedure Open_Or_Create (Container : in out
    Instance;
                          Name      : in   String;
                          Minimum_Size : in Positive);
```

Although the container contents are copied automatically to disk, and the associated file is closed on finalisation, the package still provides a `Close` procedure and a `Is_Open` function to inspect the state of the container. If the container is closed (i.e. not associated with a file), all operations but `Open_Or_Create` will raise an exception.

inc

[1] Besides the author being too lazy to make the list doubly linked from the outset.

2.2 Use

A package from the persistent container library is implemented just like one of the containers in the standard library:

```
with Persistent_Containers.Linked_List;

package Character_List is
   new Persistent_Containers.Linked_List (
       Element_Type => Character);
```

Using the list type declared through the instantiation above, we first declare a container object (`List`) and associate it with a file:

```
   List : Character_List.Instance;
begin
   List.Open_Or_Create (Name => Name,
                        Minimum_Size => Minimum_Size);
```

Then we can check if the list is empty:

```
   if List.Is_Empty then
```

And append objects if it is:

```
      Insert_Test_Data :
      for C of Test_Data loop
        List.Append (New_Item => C);
      end loop Insert_Test_Data;
```

We can manipulate the objects contained in the list:

```
   ASCII_Caesar_Code :
   for C of List loop
     C := Character'Succ (Character'Succ (
         Character'Succ (C)));
   end loop ASCII_Caesar_Code;
```

If we print the contents of the list after updating them;

```
   Iterate :
   for C of List loop
     Ada.Text_IO.Put (C);
   end loop Iterate;
```

then the output will be different (shifted three character values) every time we run the program:

```
% ./bin/example
Ghfhpehu#43wk#4;48
% ./bin/example
Jkikshkx&76zn&7>7;
% ./bin/example
Mnlnvkn{):9}q):A:>
```

The program does not have to do anything to persist the container before it stops.

The full source text of the example described in this section can be found in appendix A.

3 Implementation using memory-mapped files

3.1 Memory-mapped files

To understand memory mapped files, we can start with a quote from the POSIX specification of the function "mmap" [5]:

> The mmap() function shall establish a mapping between a process' address space and a file…

The actual copying of data between disk and RAM is handled by the operating system. Essentially the mapped file is assigned as swap space to its part of the process' address space. This gives us the possibility of saving some copying between disk and RAM; if the operating system for example already has "swapped" the file to disk, saving the data has zero cost – they are already in the file.

The big value of using memory mapping is this saving in physical copying of data between disk and RAM.

The cost of using memory mapping is that we can't handle objects containing absolute memory addresses (such as `System.Address` and access types). Other persistent implementations have the option of "flattening" structures of objects using access types for inter-object reference.

The POSIX specification of "mmap" gives us an implicit guarantee that the mapped file will contain an exact copy of the process memory once "unmap" has been called (or the program has stopped).

Although this implementation is using the Ada POSIX API, it is likely that memory mapping implementations in non-POSIX operating systems will work equally well. According to [6] "Most modern operating systems or runtime environments support some form of memory-mapped file access", so even if your target platform isn't POSIX compatible, it is likely that the technique can be used without too many modifications.

3.2 Relatively addressed, persistent heap

Between the memory mapped files and the persistent containers, there is a persistent heap addressed with relative addresses such that it does not matter where in the virtual memory the backing file is mapped to.

The `Persistent_Heap` package interfaces with the POSIX API to map and unmap the backing file. It contains a genering package, parametrised with an `Element_Type` for allocating objects on the heap, accessing the "root object" on the heap, and turning relative heap addresses into Ada 2012 style reference objects with an `Implicit_Dereference` aspect.

The details of the interfacing with the POSIX API are equivalent to the description in [3], with the major difference being that we don't ask to have the file mapped to a specific address.

3.3 Persistent containers

The demonstration implementation of a persistent linked list container primarily differes from any other linked list implementation written in Ada in how **new**, **.all**, **access** and `'Access` have been substituted with the equivalent operations and types from the `Persistent_Heap` package; `Operations.Allocate`, `Operations.Element`, `Operations.Reference_Type`.

As an example, we can look at the `Prepend` procedure:

```
procedure Prepend (Container : in out Instance;
                   New_Item : in Element_Type) is
begin
   if Container.Heap.Is_Open then
      declare
         New_Node : constant Node_Operations.
             Reference_Type
           := Node_Operations.Allocate (Container.
               Heap);
      begin
         New_Node := Node_Type'(Element =>
             New_Item,
                                Next  => Header (
                                    Container).First);
         Header (Container).First := New_Node.
             Address;
         Header (Container).Length := Header (
             Container).Length + 1;
      end;
   else
      raise Constraint_Error with "Prepend:
          Container has no file backing.";
   end if;
end Prepend;
```

`New_Node` is a `Node_Operations.Reference_Type`, which is equivalent to an **access** `Node_Type` type.

`Node_Operations.Allocate` allocates a `Node_Type` object on the indicated persistent heap.

`New_Node.Address` returns the heap-relative address of `New_Node`, as we can't just store the absolute address stored in the reference type.

The observant reader will have noticed that the implementation not yet includes the locking required to ensure safe use of the containers.

4 Comparison with other techniques

4.1 Speed

To test the actual impact of this technique, two test programs have been made. Both of them create or load a linked list of characters, and then manipulate it. The only difference between the two programs is which persistence implementation they use.

As a baseline I use the GNAT implementation of `Ada.Containers.Doubly_ Linked_Lists`, with a manual persistence implementation using `Ada.Streams.Stream_IO` and the `'Read` and `'Write` attributes of the list implementation.

The baseline is compared with the described `Persistent_Containers. Linked_List` implementation.

Three timing experiments have been performed:

Experiment	Baseline	Persistent containers
$I + M + W$	0.685	1.000
$L + W$	0.376	0.004
$L + M + W$	0.705	0.390
$M = (L + M + W) - (L + W)$	0.329	0.386

Table 1: Wall clock execution times. Each experiment is performed ten times, and the lowest and highest value is removed before an averaged is calculated. Finally the values are normalised so the highest value is 1.

I+M+W Insert test data into a linked list, modify it and write it to persistent storage.

L+W Load an existing linked list from persistent storage (disk) and write it again.

L+M+W Load an existing linked list from persistent storage (disk), modify it and write back to persistent storage.

The results from the experiments are summarised in table 1.

The first observation we can make is that neither of the implementations is consistently faster.

In pure, measured input+output performance the persistent containers win by an incredible margin. This is because of how the operating system implements memory mapping. The actual copying of data from disk to RAM only happens once the application attempts to access the mapped memory area. Similarly data are only copied from RAM to disk if the application has written to the mapped memory area, and even then only in case of swapping or once the program stops. One way of demonstrating this (see figure 1 for actual measurements) is to run the "L+W" example with larger persistent data sets. The baseline timing will scale linearly with the size of the data set, while the persistent containers timing will be constant.

The calculated times for modifying the linked list (the Caesar code operation shown in section 2.2) are roughly equivalent, with the persistent containers being slightly slower. I expect that the difference would be even larger, if the persistent containers included all the same checks as the baseline container. Considering that the persistent containers are using relative addresses, it is not at all unreasonable that they are a bit slower executing in-memory operations.

It looks like the implementation for inserting objects into the persistent container is unreasonably slow. It may be because of the number of operations on relative addresses, or because the implementation has been written without special considerations for performance.

The performance balance between stream-based and memory map-based persistence lies in the cost of reading and writing the whole container versus the relative addressing cost of each operation on the container. If an application is to run for a long enough time, the per-operation cost will out-weigh the input-output cost, making a persistence implementation based on streams preferable.

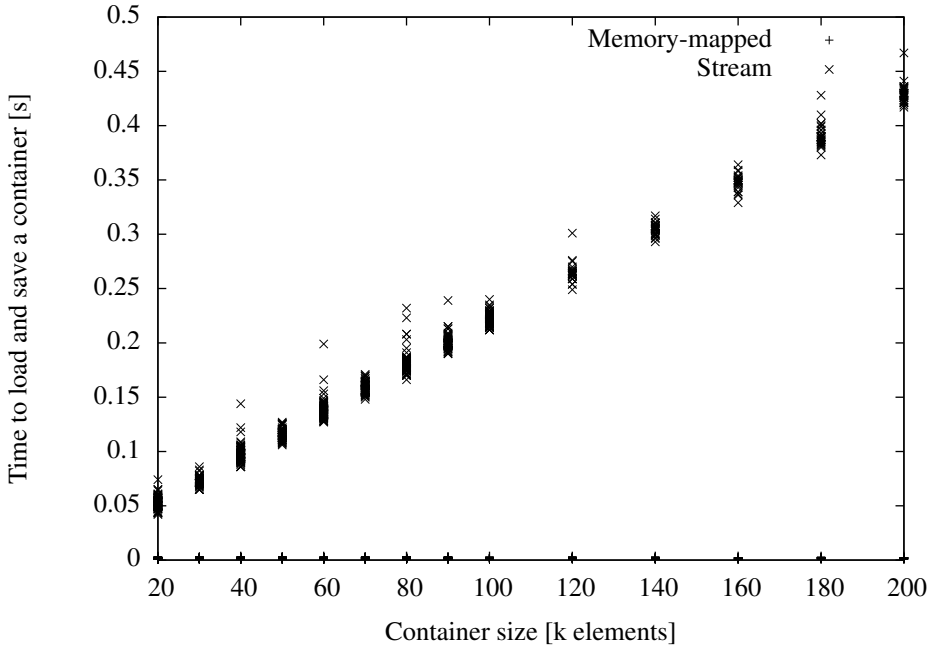

Figure 1: Measured time to load and save the contents of a container depending on the size of the data set and the persistence implementation.

4.2 Persistence manager

Using memory maps to implement persistence moves a bit of the responsibility from the application to the operating system.

One could claim that this reduces the risk of loosing data, since the operating system will take care of saving the persistent objects, if the application dies[2]. The down-side of this is that the application may die while the persistent data are in an inconsistent state, leaving the data inconsistent for the next time the application is executed. A safe implementation should therefore either maintain the persistent data constantly in a consistent state, or maintain a persistent flag, which indicates if the data are consistent or not.

4.3 Shared data

Memory maps can be shared between several processes, such that several instances of the same program can operate on the same persistent data structure. Although this requires that the programmer implements the appropriate locking using primitives supplied by the operating system[3], it is still an improvement over stream-based and other load-work-store type persistence implementations.

4.4 System calls

System calls have a large impact on the performance of applications on practically all architectures, since the CPU has to switch from the application/user context to an operating system context.

Using memory maps reduces the number of system calls needed to implement persistence to a fixed number per execution of the application, no matter how much data is being stored[4].

Implementing persistence using serialisation (streams) will result in a number of system calls which will scale linearly with the number of objects being stored. Inserting a buffering stream between the serialisation routine and the operating system, will reduce the number of system calls. With a careful implementation a buffering stream may even use as few system calls as using memory-mapped files.

4.5 Virtual memory

To understand the performance of I/O implementations on a modern operating system, it is necessary to remember that modern operating systems work with the concept of virtual memory. Virtual memory is **not** the same as RAM. Virtual memory should rather be seen as a unified address space, where the operating system freely moves the actual data around between disk (swap), RAM and CPU caches. At the same time, the operating system maintains RAM caches with parts of files. Each process has its own virtual memory, and when data are copied from an operating system controlled resource, such as a disk, to a process, there is a performance cost since the operation requires both a context switch and moving data around.

Virtual memory, disk based swap space, and RAM cached files make it hard to make an exact estimate of how large a volume of data is copied between disk and RAM. What we can do is estimate the minimal volume of data copied around. For a traditional persistence implementation it is

[2]The guarantee that the mapped file will contain an exact copy of the process memory once “unmap” has been called (i.e. once the program has stopped) is only implicit in the POSIX specification of “mmap” [5].

[3]Ada protected objects cannot be shared in this way, if one wants their semantics to be preserved.

[4]Extending the persistent storage will require some system calls.

$O(N)$ whereas the implementation presented here is $O(1)$, since the only data the operating system is required to copy is the fixed size head of the persistent storage pool file. In practise we will of course expect the process to access some of the objects in the persistent storage pool, and then they will have to be copied as well. But since we use a memory map, the whole process of managing which parts of the persistent storage pool are in RAM, and which are on disk is handled by the operating system – which has algorithms tuned through long experience to do this as efficiently as possible.

4.6 Storage format stability

When a program is recompiled, the layout of data types, type tags, etc. may change. Since Ada uses name based type equivalence, this makes sense. Unfortunately this (and name based type equivalence) will make a persistent storage pool from one version of a program unusable for another version of the program, such that programs cannot rely on this technique for long-term storage. For long-term storage – i.e. data which should persist beyond the life-time of a specific version of a program – it is still necessary to use a documented, implementation-independent storage format. The `Persistent_Heap` package, and thus also the persistent containers built on top of it, has checks to ensure that the source text version of the library using a persistent heap/container matches the version which created it.

Implementing persistence using serialisation and streams, does not automatically solve the problem of saving in an implementation-independent file format, but it is probably easier to implement that way than when mapping objects directly into memory-mapped files.

5 Summary

We have demonstrated a technique for implementing persistent objects using Ada 2012 style containers and memory-mapped files. The technique has been tested on Linux, but is expected to work on any Unix system without modifications. Use of the implementation on a Microsoft Windows system requires an extension of the (currently incomplete) Ada POSIX API, wPOSIX, to include memory mapping, but except for that, the implementation is expected to work unchanged.

We have shown how little a change to an application source text it is to make objects stored in a specific container persistent.

It is not safe to make access types and `System.Address` objects persistent using this technique.

The existing library requires an implementation of the POSIX Ada API to work, but this can be substituted with an explicit binding to the appropriate operating system calls.

Comparing with serialisation

Comparing the technique presented here with using serialisation to persist objects:

- The present technique handles data loading and storage significantly faster.
- The increased load/store speed comes at a cost when accessing the persistent objects.
- Serialisation can in some cases persist objects using access types, while that is never the case with the present technique.
- Serialisation requires only the standard library to work.

Comparing with persistent storage pools

Comparing the technique presented here with using persistent storage pools [3]:

- The present technique avoids the explicit use of pool allocation to make objects persistent.
- The present technique avoids the conflict with address space layout randomisation which is inherent in the use of absolute addresses in persistent storage pools.
- The present technique is slower, as it has to dereference relative addresses.

6 Conclusion and future work

Although the presented technique may be interesting in some cases, it looks like it – in its current form – has too many drawbacks to be generally usable as it is.

There are two obvious steps, which together will improve the benefit of using the presented technique in the areas of safety, reliability and portability:

- Substitute the `POSIX.Memory_Mapping` backing with a `Ada.Streams.Stream_IO` backing in the persistent container library, using `Ada.Finalization` to ensure that the container contents are stored when a persistent container goes out of scope.
- Extend AdaControl with rules to check if access types, `System.Address` objects, or objects related to tasking and synchronisation are written to a stream.

References

[1] Card, M.P. (1997), *Why Ada is the right choice for object databases*, CrossTalk 9–13

[2] ISO/IEC JTC 1/SC 22/WG 9 Ada Rapporteur Group: Ada Reference Manual – ISO/IEC 8652:2012(E). `http://www.adaic.org/ada-resources/standards/ada12/` (December 2012)

[3] Andersen, J.S. (2010), *An Efficient Implementation of Persistent Objects*, In: Reliable Software Technologiey – Ada-Europe 2010. Volume 6106/2010 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg 265–275

[4] Crawley, S., Oudshoorn, M. In: Orthogonal persistence and Ada

[5] The Open Group (2004), *MMAP(2)*. Issue 6 edn.

[6] Wikipedia (2014), Memory-mapped file — wikipedia, the free encyclopedia [Online; accessed 16-January-2015].

A Example source text

The full source text of the example used in section 2.2:

```
with Ada.Text_IO;

with Character_List;

procedure Example is
   Name      : constant String := "iterate.list";
   Test_Data : constant String := "December 10th
       1815";
   Minimum_Size : constant := Test_Data'Length;

   List : Character_List.Instance;
begin
   List.Open_Or_Create (Name => Name,
                        Minimum_Size => Minimum_Size);

   if List.Is_Empty then
      Insert_Test_Data :
      for C of Test_Data loop
        List.Append (New_Item => C);
      end loop Insert_Test_Data;
   end if;

   ASCII_Caesar_Code :
   for C of List loop
     C := Character'Succ (Character'Succ (
          Character'Succ (C)));
   end loop ASCII_Caesar_Code;

   Iterate :
   for C of List loop
     Ada.Text_IO.Put (C);
   end loop Iterate;
end Example;
```

A Task-Based Concurrency Scheme for Executing Component-Based Applications

Francisco Sánchez-Ledesma, Juan Pastor, Diego Alonso and Bárbara Álvarez
Division of Systems and Electronic Engineering (DSIE). Universidad Politécnica de Cartagena, Campus Muralla del Mar, E-30202, Spain; email: juanangel.pastor@upct.es

Abstract

This paper describes a flexible development approach for component–based applications with real–time requirements, which enables the performance of schedulability analysis of the resulting application. The work described in this paper is part of a more general approach, and as such it focuses on the design of a concrete part of the approach. Specifically, we describe a task-based concurrency scheme for executing component-based applications, the deployment model that enables us to configure the application execution as well as some examples of the performance of schedulability analysis of the resulting application. The aforementioned deployment model provides the approach with great flexibility, since it enables developers to generate and test different deployments of the same architecture, without modifying it, while at the same time it enables us, as the designers of the approach, to have better control over the resources and facilities required to execute the application, which is mandatory in embedded systems.

1 Introduction

Real-time (RT) systems possess specific characteristics that make them particularly sensitive to the architectural decisions made in the course of their construction. Concurrency design, task scheduling, distributed communication, etc. need to be addressed as soon as possible. However, it is not always possible to test them in the early development stages. Particularly, RT scheduling analysis cannot be performed until the final code is nearly finished and the execution platform has been selected. In case the application does not meet its timing requirements, it can be necessary to re-implement it, thereby increasing the development time and cost. Concurrent programming concepts such as thread, mutex, message, etc. are the common design units in RT systems, since they are also the analysis units. Despite being suitable for performing temporal analysis, they cannot be easily combined or composed in order to build new applications, since usually thread code and thread interaction are application specific.

Architectural software components [**?**] are self-contained units that encapsulate their state and behaviour, that communicate by sending messages through their ports, and that have only explicit context dependencies. They are normally used as the building blocks to model the application architecture, since the abstractions they provide are better suited for this purpose than those provided by concurrency. However, the design concepts that make components very suitable for application construction and code reuse, hinder the performance of schedulability analysis, since there is not a clear mapping between those concepts (i.e. port, interface, service, etc.) and concurrency concepts (i.e. thread, mutex, message, etc.). Typical examples of such mappings are:

- Component models that directly translate components into processes and that use a middleware for message exchange among them. These models provide developers with great flexibility at design time, but penalizes system performance because of the overhead introduced by the middleware. Schedulability analysis is hard to perform, since the developer must know both the threads that are created inside each process (component) and used by the middleware, as well as their timing properties.
- Component models where components are purely passive entities that are invoked sequentially by a single-threaded run-time suffer from "scant concurrency", since the application is normally executed by a cyclic executive. Despite being absolutely predictable they are very fragile in the sense that if the system needs to be modified, a task that before would fit in the slot, may now exceed it [1].

On the other hand, current object-oriented languages and frameworks provide mechanisms and libraries to flexibly manage concurrency in applications, like *java.util.concurrent*, *std::async* in C++ 11, or *android.os.AsyncTask*. In these models, the programmer enqueues the code he wants to be executed, while a pool of worker threads is in charge of dequeuing and executing them concurrently, returning the computed values by means of future objects. This is a very powerful, expressive, flexible, and easy to use model for concurrent execution. But this model has two main drawbacks from our point of view: it has a lower abstraction level for system modelling than architectural components, and its behaviour is not predictable, because worker threads dequeue and execute the activities as soon as there is one available activity and one idle thread. Therefore, it cannot be directly used in RT systems, but instead it must be slightly modified in order to make it more predictable.

The work described in this paper is part of a more general approach, entitled C-Forge [2], where programmers model

applications using architectural components whose behaviour is defined by means of state-machines with orthogonal regions. An object-oriented framework, FraCC [3, 4], provides the execution environment for the resulting application. The execution model is based on a modification of the task model just mentioned, so that schedulability analysis can be performed. In addition, FraCC provides a deployment facility that separates application architecture from its deployment in nodes, processes and threads. This separation allows developers to generate and test different deployments of the same architecture, without modifying it, while at the same time it enables us to have better control over the resources required to execute the application. Given the differences existing between the concepts of each domain (components and concurrency), C-Forge uses the *Model-Driven Software Development* paradigm [5] and its associated technologies to support the whole process. An example of the usage of C-Forge applied to underwater vehicles and the results of the schedulability analysis is described in [6].

This paper describes the task-based concurrency scheme we have developed for executing component-based applications, the deployment model that enables us to configure the application execution as well as some examples of the performance of schedulability analysis of the resulting application. The rest of the paper is organized as follows. Section 2 compares the described approach with other similar approaches. Section 3 briefly describes the task-based concurrency scheme and its main properties. Section 4 illustrates the flexibility of the approach by defining and testing some examples, while section 5 outlines the conclusions and future research lines.

2 Related work

Given the number of available component models [7], we will focus the rest of the discussion on those aimed to design software for RT systems, their concurrency capabilities and schedulability analysis.

ProCom [8] is the successor of SaveCCM. ProCom is integrated in an MDSD toolchain, which provides C++ source code generator and analysis capabilities, like worst-case execution time. ProCom defines two layers: the upper and the lower layer. The former allows developers to define large-grained components, i.e. subsystems, which are active and which communicate using message passing. The latter consists of basic functional components, which are interconnected inside subsystem, and which are passive and only activated by some external entity. Thus, components are active or passive depending on their size, but smaller components are always passive, independently of their complexity. In C-Forge, all components are active, but this does not mean that they require their own thread.

The *Architecture Analysis & Design Language* (AADL) [9] focuses on the modelling and analysis of the application architecture, both on the software and hardware platform. AADL defines components as the kind of elements that can be used to compose the software and the hardware. AADL does not support the notion of software component, as stated in the introduction, or changing the concurrency of the application, though it does support many analysis types, including schedulability analysis. There are several generators of Ada/C/C++ source code for implementing the application.

RUBUS [10] is a component model for RT systems that supports expressing timing requirements and properties on the architectural level, so that they can be later analysed. It provides schedulability analysis, distributed end-to-end response times, and overall stack analyse of the shared stacks, among others. It does not however model component behaviour, but it is added by the programmer. The execution semantics of software components (implemented as functions) is started based on an input-trigger, then read data on data in-ports, then execute the function, afterwards write data on out-ports, and finally activate the output trigger that will turn on the next connected components. RUBUS does not support concurrent execution.

The *CHESS* project [11] developed a MDSD toolchain for cross-domain modelling of RT embedded systems, that allocates distinct concerns to distinct views. It has been defined as a UML profile, including tailored MARTE profile and others OMG standards. Component behaviour can be defined with state-machines, other standard UML diagrams, as well as the *Action Language for Foundational UML* (ALF, http://www.omg.org/spec/ALF/). The Deployment view models the target execution platform, and software to hardware components allocations. The Analysis view supports Failure Mode Effects & Criticality (FMECA), Failure Mode and Effect Analysis (FMEA), Fault Tree Analysis (FTA), as well as schedulability analysis. CHESS also has generator to Ada/C/C++ /Java source code. Among the reviewed component models, CHESS is perhaps the most similar approach to C-Forge. C-Forge focus on a single way to model component behaviour and manage concurrency, which makes it easier to generate and compose code.

The Real-time Container Component Model (RT-CCM) [12] proposes a methodology for the design of component-based applications with hard real-time requirements. RT-CCM is a aimed at making the timing behaviour of applications predictable, and is inspired in the Lightweight CCM specification with some extensions. The added mechanisms also enable the application designer to configure this scheduling without interfering with the opacity typically required in component management. From the analysis of this model the application designer obtains the configuration values that must be applied to the component instances and the elements of the framework in order to make the application fulfil its timing requirements. However, RT-CCM considers components as black-boxes, while our proposal considers them as white-boxes, with their behaviour modelled by means of state-machines.

Summarizing, our approach revolves around the following reasons. Firstly, it is mandatory that the number of threads that execute the application, as well as their timing properties (mainly, computational load and period), are known in order to be able to perform a schedulability analysis. Secondly, in order to maintain the coherence between the design model (i.e., the components that define the application architecture) and the concurrency model, this data must be

somehow present in the former, so that the latter can be partially derived from it, and then completed by the developer if needed. These two reasons imply that component models that are purely structural, that is, that only provide primitives for defining the external component shell and its ports, cannot be used for this purpose. There are two viable alternatives to overcome this limitation: (i) to enhance a purely structural component model with the meta–data required to partially derive concurrency characteristics, or (ii) to enable the developer to define component behaviour together with timing requirements. The most important drawback of the first approach is that it is very difficult to assert that the component implementation is coherent with the meta–data that describes its concurrency characteristics and timing properties. We decided to follow the second approach.

3 Task-based concurrency scheme

As said in the introduction, he work described in this paper is part of a more general approach [2], where programmers model applications using architectural components whose behaviour is defined by means of state-machines with orthogonal regions. State-machines do not only model the lifecycle of components, but also enable modelling how components react to messages it receives from other components, to the results of internal computations, as well as to the passage of time. Communication among components only takes place through their ports, and is message-based, asynchronous without response. This mechanism does not only makes it possible to implement any other communication scheme as required, but also decouples component communication, since it does not allow blocking calls. As a good consequence, synchronization and message dependencies must be explicitly modelled in state-machines, which facilitates reviewing and reasoning about the component behaviour.

Regions constitute a very appropriate link between the architecture and concurrency domains. On the component domain, a region defines a part of the whole component behaviour, while on the concurrency domain, a region is assigned to the thread that will execute it. On the component domain, the states contained in a region have been enriched with properties that allow developers to define their timing constraints (mainly execution time, and period or inter-arrival time), while on the concurrency domain the thread's timing properties are derived from those of the states contained in the regions assigned to it. Regions represent computational units of work, since they contain the activities that encapsulate the code that must be executed by the component depending on its internal state. Though a region can contain many activities, only the activity associated to the active state can be executed.

The concurrency model we have developed in order to organize and control region execution in threads is based on a modification of the task-based scheme used in systems like *java.util.concurrent*, *std::async* in C++ 11, *Grand Central Dispatch* in iOS, or *android.os.AsyncTask*, to mention a few. In this model, the main thread enqueues the activities it wants to be asynchronously performed, while a pool of worker threads is in charge of dequeuing and executing them concurrently, returning the computed values by means of future objects. This is a very powerful, expressive, flexible, and easy to use model for concurrent execution, but its behaviour is not predictable, because worker threads dequeue and execute the activities as soon as there is one available activity and one idle thread. As such, it cannot be directly used in RT systems, but instead it must be slightly modified in order to make it more predictable:

- Make the computational load of worker threads static, decided by the user at development time instead of by the system at execution time.
- Convert the main thread into a "normal", worker thread, since there is not such a thing as "a main component" in C-Forge.
- Let the developer decide how many (worker) threads execute the application.
- Create a cyclic executive inside each thread in order to schedule region execution.

The proposed task-based concurrency scheme for executing component-based applications start by characterizing states. States that contain one activity also define its period, deadline, worst case execution time, and activation pattern (periodic or sporadic): $St_i = (T^i_{act}, WCET^i_{act})$; data that is obtained from the application requirements. We assume that period equals deadline, and that period also model the minimum inter-arrival time in the case of sporadic activities. Starting from this data it is possible to calculate the timing properties of the regions of all components by applying equations 1 and 2. This is a pessimistic estimation, since we assume the region will be always executing the activity with the largest execution time, but it is needed in order to perform the schedulability analysis.

$$T^i_{reg} = gcd(T_{act} \in R^i) \tag{1}$$

$$WCET^i_{reg} = max(WCET_{act} \in R^i) \tag{2}$$

On the other hand, the application can be executed in a set of nodes, which represent computational units. They contain a finite set of processes, which represent the unit of resource management. Processes contain a finite set of threads, which represent the unit of concurrent execution. Components are assigned to processes and the regions of a given component can be assigned to any of the threads of the process that contains such component. This is a flexible scheme, which enables threads to execute regions contained in different components, but which does not force to assign all the regions of a component to the same thread.

Threads can be characterized by their period and their worst case execution time: $Th^i = (T^i_{th}, WCET^i_{th})$, which can be derived from the assigned regions by applying equations 3 and 4.

$$T^i_{th} = gcd(T_{reg} \in Th^i) \tag{3}$$

$$WCET^i_{th} = \sum(WCET^i_{reg} \in Th^i) \tag{4}$$

A cyclic executive scheduler is created inside each thread in order to control the execution of the regions assigned to it. Given that the assignment of regions to threads is static and is made at design time, it is possible to automatically calculate the parameters needed by the cyclic executive, primary cycle (H) and secondary cycle (T_s), and build the execution table from such assignment. The primary cycle (H) can be calculated by means of equation 5, while the secondary cycle coincides with the thread period, calculated by means of equation 3.

$$H^i = lcm(T_{reg} \in Th^i) \tag{5}$$

It should be highlighted that FraCC does not give any guidance as to the number of threads that have to be created or how regions should be assigned to them, but rather it provides the necessary mechanisms to enable developers to choose the appropriate heuristic methods, like the ones defined in [13], for instance. Both the number of threads as well as the allocation of regions to them can be done arbitrary, but the main objective should be to "*ensure application schedulability*". Two heuristics we normally use are to assign to the same thread regions that have similar periods, or that have states which activities communicate with each other.

3.1 Schedulability Analysis

A deployment model in C-Forge enables developers to set the application distribution in computational nodes, as well as the number of processes and threads in which the application will be run, as described previously. This organization makes it possible to perform schedulability analysis of a given application deployment. The deployment model provides great flexibility, since it does not impose a fixed relationship between component and processes/threads, but rather allows developers to define it, within certain limits. It also enables us to better control the resources and facilities needed by the platform in order to execute the application, and use only the necessary ones, as well as the performance of RT schedulability analysis. For instance, if all the application components run in the same node, no middleware is really needed, and thus lighter mechanisms, like shared memory, can be used instead for message exchange.

Cheddar [14] is a RT scheduling tool, designed for checking task temporal constraints of a RT system. In order to perform the schedulability analysis, Cheddar requires the number of tasks, their timing properties (mainly wcet and period) and the number of shared resources of the application. *Thread*s of the deployment model are directly transformed into Cheddar tasks, but shared resources must be derived from the deployment model, according to the buffer structures implemented in FraCC, as described in [3]. It must be highlighted that shared resources do not use synchronization primitives, only mutual exclusion, due to the fact that communication among components is only asynchronous. This makes it possible to bound blocking times.

According to the memory structure, only the buffers are candidates to be structures shared among threads. Among the generated buffers, only those that hold messages sent or processed by activities contained in regions assigned to different threads need to be protected from concurrent access. Buffers that hold messages produced or consumed by activities contained in regions assigned to the same thread need not be protected, since they will be accessed sequentially by activities. These shared buffers use the immediate ceiling priority protocol. It must also be noted that there is only one active state per region, and thus only one activity per region will access these buffers. All the needed information can be derived from the architectural and deployment models.

Table 1: Regions' calculated timing properties

Region	Period (ms)	WCET (ms)
R1	10	0.5
R2	20	1
R3	2	0.5
R4	40	0.8
R5	20	1
R6	2	1

In case the schedulability analysis concludes that the application is not schedulable, the developer can first generate new deployment models, mainly by changing the number of threads and the assignment of regions to threads. If the applications continues to be not schedulable, he/she has to start modifying the architecture, mainly by changing the algorithms used in the activities to faster ones, or by relaxing the timing constraints of the states. The last option if none of the previous generates a schedulable application is to change the components themselves, and thus the application architecture.

4 Sample Application

In order to illustrate the system and execution models described in the previous section, as well as the schedulability analysis, the sample application shown in figure 1 will be used. As can be seen, it comprises three components and six regions, which timing properties are also depicted in the figure. We assume that the application will run in one node and one process.

4.0.1 Region characterization.

The timing properties of the regions are derived from their activities by applying equations 1 and 2 as shown in Table 1.

4.0.2 Region to threads assignment.

Regions can be assigned to threads in an arbitrary way. A possible thread scheme considers four threads to execute the application, with the following assignment: $Th1 = \{R2\}$, $Th2 = \{R1, R4, R5\}$, $Th3 = \{R3\}$, and $Th4 = \{R6\}$. The following subsection will present some deployment examples in which we change this assignment and the results of the Cheddar analysis.

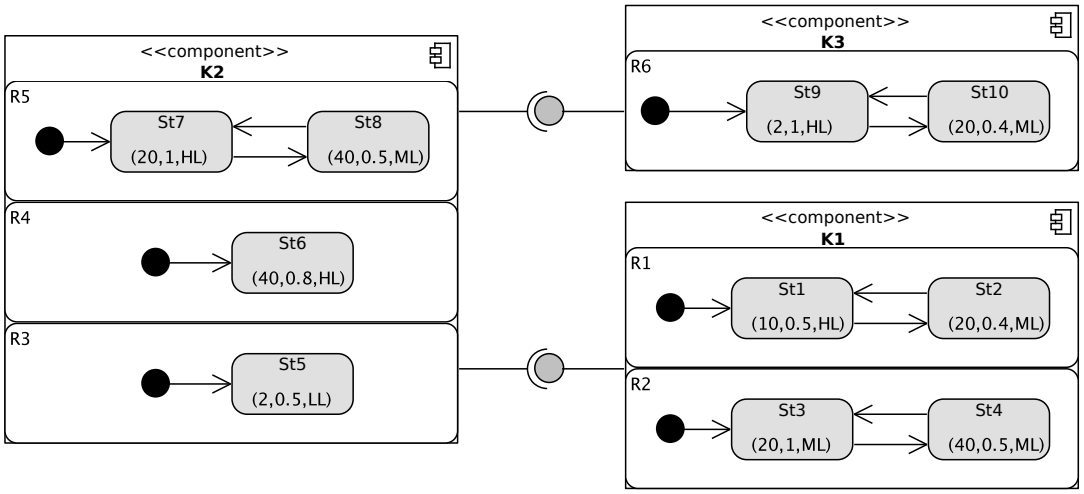

Figure 1: Sample application architecture with timing properties.

Table 2: Threads' calculated timing properties

Thread	Region/s	Period (ms)	WCET (ms)	Priority
Th1	R2	20	1	4
Th2	R1, R4, R5	10	2.3	3
Th3	R3	2	0.5	2
Th4	R6	2	1	1

4.0.3 Threads characterization.

Given this assignment, the timing properties of the thread can be calculated by applying equations 3 and 4, as shown in Table 2. The rate monotonic algorithm is used to calculate the concrete priority level. The lower the priority number (Pr), the higher the thread priority.

4.0.4 Scheduling regions inside threads.

Threads $Th1, Th3$, and $Th4$ do not need to schedule the regions inside them because they only have one region each, but thread $Th2$ does need to schedule regions $R1$, $R4$, and $R5$. To schedule these regions we need to calculate the primary and secondary cycles, as well as to build the scheduling table. Primary cycle is calculated by applying equation 3: $H_2 = lcm(T_{R1}, T_{R4}, T_{R5}) = lcm(10ms, 40ms, 20ms) = 40ms$, while the secondary cycle coincides with the thread period, $T_{s2} = 10ms$. Thus, the scheduling table will have four secondary cycles of $10ms$ each:

$t = 0ms$ Executes $R1$, $R4$ and $R5$
$t = 10ms$ Executes $R1$
$t = 20ms$ Executes $R1$ and R_5
$t = 30ms$ Executes $R1$

4.1 Deployment Examples

Table 3 shows the results of the Cheddar analysis for the sample deployments we describe below. The default deployment model created by the tool, deployment 1, defines one node with a single process hosting just one thread. All components are assigned to this process, while all regions of the components are assigned to such thread. Given the periods and worst execution times of the components regions, it is clear that the application resulting from the default deployment is not schedulable.

In deployment 2 the developer has defined three threads, one for executing each component. At a glance, it is possible to determine that the application is again not schedulable. Note that the period of thread 3 is lesser than its WCET. The developer can change the regions assignment, as it is shown in deployment 3, in order to reduce the WCET of thread 3. This new deployment is now schedulable. Deployment 4 shows the case where all the components' regions have been assigned to different threads, resulting, for this example, in the best of the four deployments from the point of view of processor usage.

5 Conclusions

This paper describes a flexible development approach for component–based applications with real–time requirements, which provides developers with enough control over the concurrency characteristics of the application execution so that schedulability analysis can be performed. These objectives have been achieved by means of (i) defining a component model that includes structure and behaviour; (ii) establishing a clear separation between these concerns, decoupling the structural elements from the behavioural and the algorithmic ones; (iii) defining a clear and consistent association between the elements of the system and execution models through a deployment model. The approach is supported by a model-driven toolchain developed in Eclipse (C-Forge).

The explicit modelling of component behaviour by means of state-machines with orthogonal regions offers several advantages, namely it enables developers to describe the temporal requirements at the architectural level; orthogonal regions explicitly reflect the concurrent nature of the component behaviour; regions have proven to be an excellent way to link the architecture and concurrency domains, since on the component domain regions define a part of the whole component behaviour, while on the concurrency domain they define the unit of computational work assigned to a thread. The concurrency scheme we developed in order to organize and control

Table 3: Summary of the four considered deployments and the results of the Cheddar schedulability analysis

Deployment 1 (T, WCET)	Deployment 2	Deployment 3	Deployment 4
Thread1 (T=2, WCET=4.8) Reg1 (10, 0.5) Reg2 (20, 1.0) Reg3 (2, 0,5) Reg4 (40, 0.8) Reg5 (20, 1.0) Reg6 (2, 1.0)	**Thread1 (T=10, WCET=1.5)** Reg1 (10, 0.5) Reg2 (20, 1.0) **Thread2 (T=2, WCET=1)** Reg6 (2, 1.0) **Thread3 (T=2, WCET=2.3)** Reg3 (2, 0,5) Reg4 (40, 0.8) Reg5 (20, 1.0)	**Thread1 (T=10, WCET=1.5)** Reg1 (10, 0.5) Reg2 (20, 1.0) **Thread2 (T=20, WCET=1.8)** Reg4 (40, 0.8) Reg5 (20, 1.0) **Thread3 (T=2, WCET=1.5)** Reg3 (2, 0,5) Reg6 (2, 1.0)	**Thread1 (T=10, WCET=0.5)** Reg1 (10, 0.5) **Thread2 (T=20, WCET=1.0)** Reg2 (20, 1.0) **Thread3 (T=2, WCET=0.5)** Reg3 (2, 0,5) **Thread4 (T=40, WCET=0.8)** Reg4 (40, 0.8) **Thread5 (T=20, WCET=1.0)** Reg5 (20, 1.0) **Thread6 (T=2, WCET=1.0)** Reg6 (2, 1.0)
Cheddar analysis results: ***Feasibility test based on the processor utilization factor:*** – Processor utilization factor with deadline is 2.4 – In the pre-emptive case, with RM, cannot prove that the task set is schedulable: processor utilization factor is more than 1.0 ***Feasibility test based on worst case task response time:*** Processor utilization exceeded: cannot compute bound on the response time with this task set.	**Cheddar analysis results:** ***Feasibility test based on the processor utilization factor:*** – Processor utilization factor with deadline is 1.52 – In the pre-emptive case, with RM, cannot prove that the task set is schedulable: processor utilization factor is more than 1.0 ***Feasibility test based on worst case task response time:*** Processor utilization exceeded: cannot compute bound on the response time with this task set.	**Cheddar analysis results:** ***Feasibility test based on the processor utilization factor:*** – Processor utilization factor with period is 0.99 – 200 µs are unused in the base period. – In the pre-emptive case, with RM, the task set is schedulable. ***Feasibility test based on worst case task response time:*** Bound task response time: Thread2 ⇒ 19800 µs Thread1 ⇒ 6000 µs Thread3 ⇒ 1500 µs	**Cheddar analysis results:** ***Feasibility test based on the processor utilization factor:*** – Processor utilization factor with period is 0.92 – 3200 µs are unused in the base period. – In the pre-emptive case, with RM, the task set is schedulable. ***Feasibility test based on worst case task response time:*** Bound task response time: Thread4 ⇒ 15800 µs Thread2 ⇒ 10000 µs Thread5 ⇒ 6000 µs Thread1 ⇒ 2000 µs Thread3 ⇒ 1500 µs Thread6 ⇒ 1000 µs

region execution in threads revolves around a modification of the thread-pool design, where regions are the units of work; developers define at design time both the number of threads that execute the application, as well as their computational load, by assigning the regions they will execute; and a cyclic executive inside each threads manages region execution. The regularity of this scheme enables the performance of schedulability analysis, and thus its use in applications with timing requirements.

The deployment model has also proven to be essential in the approach, since it separates application architecture from its deployment in terms of nodes, processes and threads, enabling the separation of roles in the development team, as well as the rapid testing of different deployment scenarios. This model also enables us to determine the computational resources required by the application, as well as to estimate memory consumption, which is very important in embedded systems. Unlike other reviewed component models, C-Forge does not enforce a rigid association between components and processes/threads, but it can be easily configured thanks to the deployment model. It also means that C-Forge components are not forced to use a communication software for message exchange in all scenarios, but only on those where the application is distributed in more than one node.

Regarding future works, we are currently enhancing the deployment model for supporting multi-core systems, and end-to-end transactions specification, as well as automatically generating and testing different deployments, in order to find an optimum one. We are also interested in generating a less pessimistic analysis file, since we now assume that components are always executing the states with the longest computation, which cannot be possible in some cases. A more exhaustive analysis of the state-machines will enable us to make less pessimistic analysis.

References

[1] M. Ben-Ari (2006), *Principles of Concurrent and Distributed Programming*, Addison-Wesley.

[2] D. Alonso, F. Sánchez-Ledesma, P. Sanchez, J. A. Pastor, and B. Álvarez (2014), *Models and frameworks: a synergistic association for developing component-based applications*, The Scientific World Journal, pp. 1–17.

[3] D. Alonso, F. Sánchez-Ledesma, P. Sánchez and B. Álvarez (2014), *Embedded and Real Time System Development: A Software Engineering Perspective*, A flexible framework for Component based Application with Real-Time Requirements and its Supporting Execution Framework, pp. 3–22, Springer-Verlag.

[4] J. A. Pastor, D. Alonso, P. Sanchez and B. Álvarez., *Towards the definition of a pattern sequence for real-time applications using a model-driven engineering approach*, The Scientific World Journal, pp. 1–17.

[5] J. Bezivin (2005), *On the unification power of models*, Journal of Systems and Software, pp. 171–188.

[6] F. J. Ortiz, C. Insaurralde, D. Alonso, F. Sanchez and Y. Petillot (2014), *Model-driven analysis and design for software development of autonomous underwater vehicles*, Robotica, pp. 1–20.

[7] I. Crnkovic, S. Sentilles, A. Vulgarakis and M. R. V. Chaudron (2011), *A classification framework for software component models*, IEEE Trans. Software Eng., pp. 37(5):593–615.

[8] A. Vulgarakis, J. Suryadevara, J. Carlson, C. Seceleanu and P. Pettersson (2009), *Formal semantics of the procom real-time component model*, Proc. of the 35^{th} Euromicro Conference on Software Engineering and Advanced Applications, pp. 478–485, IEEE.

[9] P. Feiler and D. Gluch (2012), *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Languages*, Addison Wesley Professional.

[10] K. Hanninen et al (2008), *The rubus component model for resource constrained real-time*, International Symposium on Industrial Embedded Systems, pp. 177–183, IEEE.

[11] A. Cicchetti et al (2012), *Chess: a model-driven engineering tool environment for aiding the development of complex industrial systems*, in Proc. of the 27^{th} IEEE/ACM International Conference on Automated Software Engineering, pp. 362–365, ACM Press.

[12] Patricia López Martínez, L. Barros and J. M. Drake (2013), *Design of component-based real-time applications*, Journal of Systems and Software, pp. 86(2):449–467.

[13] P. Feiler and D. Gluch (2000), *Designing Concurrent, Distributed, and Real-Time Applications with UML*, Object Technology, Addison-Wesley.

[14] F. Singhoff, A. Plantec, P. Dissaux and J. Legrand (2009), *Investigating the usability of real-time scheduling theory with the cheddar project*, Journal of Real Time Systems, pp. 43(3):259–295.

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 in Sweden

attn. Rei Stråhle
Rimbogatan 18
SE-753 24 Uppsala
Sweden
Phone: +46 73 253 7998
Email: rei@ada-sweden.org
URL: www.ada-sweden.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