ADA USER JOURNAL

Volume 38
Number 3
September 2017

Contents

	Page
Editorial Policy for *Ada User Journal*	114
Editorial	115
Quarterly News Digest	116
Conference Calendar	132
Forthcoming Events	138
Special Contribution	
J. Tokar *"The Ada High-Integrity Rapporteur Group (HRG) Status Report"*	141
Ada-Europe 2017 Industrial Presentations	
A. Marriot and U. Maurer *"Using GtkAda in Practice"*	143
Articles	
C. Brandon, P. Chapin, C. Farnsworth, S. Klink *"CubedOS: A Verified CubeSat Operating System"*	151
N. Abid, L. Loukil, W. Ayedi, M. Abid, A. C. Ammari *"Parallelizing an Embedded Real-Time Person Matching System for Smart Cameras"*	157
Ada-Europe Associate Members (National Ada Organizations)	168
Ada-Europe Sponsors	Inside Back Cover

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

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

Original Papers

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

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

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

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

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

Proceedings and Special Issues

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

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

News and Product Announcements

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

Reprinted Articles

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

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

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

Editorial

This issue of the Ada User Journal starts with a report on the work of the High-Integrity Rapporteur Group (HRG), provided by the Chair of the HRG, Joyce Tokar, from Pyrrhus Software, USA. The HRG, under the auspices of WG9, deals with the concerns of using Ada and its development tools in high integrity systems. We intend that the work of the groups involved in the Ada standardization process becomes a recurring theme in the Journal, as it is more and more relevant to give visibility to the effort of the experts that guarantee that Ada continues to meet with the evolving needs of applications and users.

Afterwards, the issue continues with the publication of contributions from the Industrial Track of the Ada-Europe 2017 conference, with a paper by Ahlan Marriott and Urs Maurer, from White Elephant, Switzerland, providing an experience report on using GTKAda on three different operating systems.

Continuing with the technical contributions, the issue then provides an article on CubedOS, a SPARK lightweight framework for CubeSat software, written by a group of authors from Vermont Technical College, USA, which have been successfully taking Ada and SPARK into space. Finally, an article describing approaches to parallelise a real-time embedded software, by a group of authors from the National School of Engineering of Sfax, Tunisia and the King Abdulaziz University, Saudi Arabia.

As usual, the reader will find the valuable information of the News and Calendar sections, contributed by Jacob Sparre Andersen and Dirk Craeynest, their respective editors. I would also like to draw your attention to the two Ada focused events taking place next year, the International Real-Time Ada Workshop (IRTAW) and the Ada-Europe 2018 Conference. IRTAW has been one of the main drivers of the evolution of the concurrency (and now parallelism) and real-time models of Ada. Ada-Europe, the International Conference on Reliable Software Technologies, is more and more the key event on Ada and other technologies for reliable and high-integrity systems (and next year it takes place in a very beautiful city and country!). As usual, these events live from the effort and contributions of the community – please consider submitting your works and participating.

The reader will forgive me for ending this editorial with a short personal note, as this issue marks my 10th year serving as the Journal Editor. During this period, I had the pleasure to work as a member of an excellent team of volunteers, preparing 40 journal issues, which I hope to have met with your expectations and interests. I would like to apologise for any mishaps during these years, and say that we will continue thriving to provide our readers with high-quality and timely information on Ada and reliable software technologies.

Luís Miguel Pinho
Porto
September 2017
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 Organizations	116
Ada-related Events	117
Ada-related Resources	117
Ada-related Tools	117
Ada-related Products	119
Ada and Operating Systems	120
References to Publications	121
Ada Inside	121
Ada in Context	121

Ada-related Organisations

Community Input for the ARG

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 3 Aug 2017 00:45:49 -0500
Subject: Community Input for the Maintenance and Revision of the Ada Programming Language
Newsgroups: comp.lang.ada

ISO/IEC JTC 1/SC 22/WG 9 (WG 9) is responsible for the maintenance and revision of the Ada Programming Language and associated standards and technical reports. As part of the language maintenance activity, WG 9 has established a group of Ada experts as the Ada Rapporteur Group (ARG). The ARG receives input from the Ada community at large to consider for inclusion in revision to the Ada programming language standard. The WG 9 has produced a number of revisions to the language in accordance with ISO policy and to address the evolution of technology (Ada 83, Ada 95, Ada 2005, Ada 2012).

Presently, the ARG is beginning work on a revision to Ada 2012 so that ISO standardization of the new version can be completed by 2020. This is a relatively short horizon, but it ensures that the language continues to evolve, and at the same time requires that the changes to the language are evolutionary and do not present an undue implementation burden on existing compilers and users.

WG 9 requests the Ada community to submit enhancements to be considered for inclusion in the next revision of Ada. These should be sent to ada-comment@ada-auth.org as described in the Ada Reference Manual Introduction (http://www.ada-auth.org/standards/rm12_w_tc1/html/RM-0-3.html#p58). For enhancement requests, it is very important to describe the programming problem and why the Ada 2012 solution is complex, expensive, or impossible. A detailed description of a specific enhancement is welcome but not necessarily required. The goal of the ARG is to solve as many programming problems as possible with new/enhanced Ada features that fit into the existing Ada framework. Thus the ARG will be looking at the language as a whole, which may suggest alternative solutions to the problem posed by an enhancement request. For a more detailed discussion, the guidelines presented for the Ada 2005 revision (see http://archive.adaic.com/news/pressrelease/call4apis.html) can be used as the ARG requirements are little changed.

WG9 accepts enhancement requests at any time. To be considered for inclusion in the next revision of Ada, enhancement requests must be received by 15 January 2018. Suggestions received after that date may be considered if they relate to topics already under development; others will be considered only for future versions of Ada.

WG 9 has directed the ARG to focus its work on three areas of particular interest to the Ada community: additional facilities for multi-core and multithreaded programming, improved facilities for program correctness, and enhanced container libraries. There are numerous proposed enhancements in these and other areas. Some of these proposals originated with members of the ARG, and others from members of the community at large. The interested reader can find the current state of these at http://www.ada-auth.org/AI12-SUMMARY.HTML.

WG 9 encourages members of the Ada community at large to use the guidelines outlined above to provide input to WG 9 and the ARG for needed revisions and upgrades to the Ada programming language.

From: Pascal Leroy
Date: 2003
Subject: ISO Working Group asks Ada Community for Candidate APIs for Standardization
URL: http://archive.adaic.com/news/pressrelease/call4apis.html

[We're re-running this as a reminder for the community. The request is still open. —sparre]

As part of the next revision of Ada, planned for 2005, there has been a lot of interest in the Ada community for the standardization of reusable components and APIs to existing services. It is felt that such standardizations would improve the marketability of the language as well as day-to-day programmer productivity.

For most of these APIs, the proper standardization vehicle is a secondary standard (that is, a standard referencing the Ada standard, but standardized as a separate process). For relatively small APIs, inclusion in an existing annex is also an option, although this might delay the language standardization process.

The Ada Rapporteur Group (ARG) is the technical committee in charge of proposing amendments to the language to WG9, the ISO working group on Ada. While the ARG will conduct (based on input from the Ada community) the revision of the core language and annexes, it doesn't have the resources to develop proposals itself for the standardization of reusable components or APIs. The ARG will oversee the development of secondary standards, but this is best accomplished by cooperating with external groups developing the substance of such standards.

We would like to ask the Ada community to submit proposals for the standardization of APIs. Proposals should be sent to ada-comment@ada-auth.org, and should preferably have the form of an amendment AI (see http://www.ada-auth.org/cgi-bin/cvsweb.cgi/AIs/AI-00248.TXT for an example). While all input will be carefully reviewed, the ARG will act as a filter to retain only those proposals that have a sufficient level of maturity and usefulness, and will provide feedback to the authors. Criteria that will be used for evaluating the proposals include:

- Benefits of the standardization. Presumably the advantage of standardization is that it brings uniformity and portability among implementations. However, there is a significant overhead associated with a formal standardization process, so in some cases a de facto standard may bring practically the same benefits at a much lower cost.
- Usefulness of the API. APIs which have been conjured up solely for the purpose of writing a proposal, or which have been used by a very small group of users, are less likely to be generally

useful than APIs which have been available for years and have benefited from feedback from a large user base.

- Quality and precision of the proposal. At a minimum, the proposal must include a set of Ada specifications, and a semi-formal description of the semantics of each declaration, such as can be found in the annexes of the Reference Manual. A rationale showing examples of use, explaining the choices that were made, the alternatives that were considered, and why they were discarded, would also be much appreciated.
- Community consensus for the proposal. Proposals with a substantial consensus of the Ada community or the appropriate subcommunity are preferred over proposals made by an individual or small group. This is not to say that a proposal primarily authored by an individual is necessarily bad (indeed, it is likely to provide a more consistent proposal), but to encourage authors to seek input/approval from as many potential users of the API as possible.
- Portability and language usage. The definition of the API must not depend on implementation-defined characteristics of a particular compiler, although it is acceptable to require the compiler to support some Specialized Needs Annex (or part thereof). As much as possible, the API should only use the features of Ada 95 (as opposed to those that are under consideration for the 200Y amendment) although we realize that this may not be practical in some cases.
- Implementation. A publicly available reference implementation would be useful, although this is not a strict requirement, as in some cases that may cause intellectual property issues.
- Test suite. A test suite ensuring conformity to the specification should be provided at some point during the standardization process. This is especially important for standards for which no publicly available reference implementation will be available. This doesn't necessarily mean that there will be a formal conformity assessment process like there is for compilers, but it will help implementers ensure that they comply with the standard.

It is anticipated that the groups submitting proposals will keep ownership of the standard during the entire standardization process, although the ARG will provide guidance regarding that process and continuous feedback on the contents of the proposal.

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]

Swiss Ada Event

From: Ahlan Marriott <ahlan@marriott.org>
Date: Thu, 17 Aug 2017 01:25:11 -0700 (PDT)
Subject: ANN: Swiss Ada Event 21-Sep-17
Newsgroups: comp.lang.ada

Ada Switzerland is organising a half day Ada event on Thursday 21-Sep-17 at HSR in Rapperswil.

It will start at 14:00 and pause at 15:30 for a coffee break, restarting at 16:00 and finishing at 17:30 whereafter there will be an apéro for networking opportunities.

Entrance will be free and is open to anyone, not just members of Ada Switzerland.

For logistical reasons, those wishing to attend the event are cordially requested to register by sending an email to registration@ada-switzerland.ch containing their name and address and contact email.

Full details, including the preliminary program, can be obtained from the Ada-Switzerland web site: www.ada-switzerland.ch

Ada-related Resources

Ada on Social Media

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Wed Aug 30 2017
Subject: Ada on Social Media

Ada groups on various social media:

- LinkedIn:	2_650 members	[1]
- Reddit:	1_025 readers	[2]
- StackOverflow:	849 followers	[3]
- Google+:	749 members	[4]
- Freenode	77 participants	[5]
- Gitter:	52 people	[6]
- Twitter:	14 tweeters	[7]

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

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

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

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

[5] #Ada on irc.freenode.net

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

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

[See also "Ada on Social Media", AUJ 38-2, p. 70. —sparre]

Repositories of Open Source Software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Wed Aug 30 2017
Subject: Repositories of Open Source software

GitHub:	714 repositories	[1]
	421 developers	[1]
	635 issues	[1]
Rosetta Code:	635 examples	[2]
	32 developers	[3]
	0 issues	[4]
Sourceforge:	259 repositories	[5]
BlackDuck OpenHUB:	210 projects	[6]
Bitbucket:	79 repositories	[7]
Codelabs:	45 repositories	[8]
OpenDO Forge:	24 projects	[9]
	529 developers	[9]
AdaForge:	8 repositories	[10]

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

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

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

[4] http://rosettacode.org/wiki/Category:Ada_examples_needing_attention

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

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

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

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

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

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

[See also "Repositories of Open Source Software", AUJ 38-2, p. 70. —sparre]

Ada-related Tools

VTKAda

From: Leonid Dulman <leonid.dulman@gmail.com>
Date: Sat, 1 Jul 2017 02:02:41 -0700 (PDT)
Subject: Announce: VTKAda version 8.0.0 release 01/07/2017
Newsgroups: comp.lang.ada

I'm pleased to announce

VTKAda version 8.0.0 free edition release 01/07/2017

VTKAda is Ada-2012 port to VTK (Visualization Toolkit by Kitware, Inc) and Qt5 application and UI framework by Nokia.

VTK version 8.0.0, Qt version 5.9.0 open source and vtkc.dll, vtkc2.dll, qt5c.dll (libvtkc.so, libvtkc2.so, libqt5c.so) were built with Microsoft Visual Studio 2015 in Windows (WIN32) and gcc in Linux x86-64.

Package was tested with gnat gpl 2017 ada compiler in Windows 10 64bit, Debian 8.5 x86-64.

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

VTKAda you can be used without QtAda subsystem.

Qt5Ada and VTKAda for Windows, Linux (Unix) is available from

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

[See also "VTKAda", AUJ 38-1, p. 5. —sparre]

RDF Processing

From: Victor Porton <porton@narod.ru>
Date: Mon, 10 Jul 2017 02:42:29 +0300
Subject: RDF in Ada (new library released)
Newsgroups: comp.lang.ada

Raptor is a C library for working with RDF network data format (a library capable of downloading, parsing, and serializing RDF resources).

I have added (thick, object oriented) Ada bindings for Raptor (but not for Rasqal and Redland, for which I do not have time to work on).

See

https://github.com/vporton/ redland-bindings/tree/ada2012/ada

Note that the target `make dist` does not work yet. So it is a Git only distribution yet. We must get into Debian.

The API is not quite stable yet.

Test coverage is yet very partial.

I will probably write an article for Ada User Journal about how I did it in an object-oriented way.

It is a part of a big, ambitious open source project:

en.wikiversity.org/wiki/Automatic_transformation_of_XML_namespaces

Emacs Ada Mode

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Thu, 13 Jul 2017 06:52:08 -0700 (PDT)
Subject: Ada mode 5.2.2 released
Newsgroups: comp.lang.ada

Ada mode 5.2.2 is now available in GNU ELPA. See the homepage (http://www.nongnu.org/ada-mode/) for NEWS, or the project page (https://savannah.nongnu.org/projects/ada-mode) for tarball download.

A major new feature in this release; the GPS indentation engine can be used as the primary or backup indentation engine. This makes indenting while editing faster on large files, and more friendly on all files.

[See also "Emacs Ada Mode", AUJ 37-4, p. 188. —sparre]

Gnoga

From: Pascal Pignard <p.p14@orange.fr>
Date: Sun, 16 Jul 2017 13:21:51 +0200
Subject: V1.3-beta release.
Newsgroups: gmane.comp.lang.ada.gnoga

Version V1.3-beta has been released on SF GIT branch Dev_1.3:

https://sourceforge.net/p/gnoga/code/ci/dev_1.3/tree/

V1.3 will not more change for new features.

See HISTORY for features added.

V1.3 has been tested (demos, tests, tutorials) with GNAT GPL 2017 on macOS 10.11 with Safari 10.

Volunteers are welcome to test it on their own configuration.

Some testing on Windows and Linux configuration will be appreciated.

Just get last commit on https://sourceforge.net/p/gnoga, do:

```
$ git clone https://git.code.sf.net/p/gnoga/code gnoga-code
$ git checkout dev_1.3
$ make all
```

and for courageous:

```
$ make tests
```

then:

```
$ cd bin
```

and test.

Feel free to report detailed issues on this list or create tickets on SF.

From: Pascal Pignard <p.p14@orange.fr>
Date: Sun, 23 Jul 2017 13:04:36 +0200
Subject: V1.2b release.
Newsgroups: gmane.comp.lang.ada.gnoga

Version 1.2b has been released on SF GIT master branch.

https://sourceforge.net/p/gnoga/code/ci/master/tree/

Mostly this release updates Jeff Carter's demos from Github.

https://github.com/jrcarter

See HISTORY for details.

This version has been zipped on:

https://sourceforge.net/projects/gnoga/files/

Feel free to report detailed issues on this list or create tickets on SF.

Request: SOCKS5 Client Library

From: Adam Jensen <hanzer@riseup.net>
Date: Wed, 19 Jul 2017 03:27:04 -0000 (UTC)
Subject: SOCKS5 client library
Newsgroups: comp.lang.ada

Does anyone know of SOCKS5 client library written in Ada?

https://tools.ietf.org/html/rfc1928

Simple Components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 24 Jul 2017 09:54:12 +0200
Subject: ANN: Simple Components for Ada v4.22
Newsgroups: comp.lang.ada

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

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

Changes to the previous version:

- Bug fix in the package Block_Streams. Only very large transmissions were affected by it;
- Bug fix in the HTTP server implementation. The query part is now recognized when the status line contains file name.

[See also "Simple Components", AUJ 38-2, p. 72. —sparre]

Fuzzy Sets

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 25 Jul 2017 17:38:54 +0200
Subject: ANN: Fuzzy sets for Ada v5.12
Newsgroups: comp.lang.ada

The current version includes distributions of string edit, interval arithmetic and simple components packages. It provides implementations of:

- Confidence factors with the operations not, and, or, xor, +, *;
- Classical fuzzy sets with the set-theoretic operations and the operations of the possibility theory;
- Intuitionistic fuzzy sets with the operations on them;
- Fuzzy logic based on the intuitionistic fuzzy sets and the possibility theory;
- Fuzzy numbers both integer and floating-point ones with conventional arithmetical operations;
- Dimensioned fuzzy numbers;
- Fuzzy linguistic variables and sets of linguistic variables with operations on them;
- Dimensioned fuzzy linguistic variables and sets;
- String-oriented I/O is supported;
- GUI interface based on GTK+ (The GIMP Toolkit) with fuzzy set editors, truth values widgets and renderers, linguistic variables sets editors.

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

Changes to the previous version:

- Workaround GNAT 7 (20170622) bug in generic instantiation

[See also "Fuzzy Sets", AUJ 35-3, p. 157. —sparre]

Cortex GNAT RTS

From: Simon Wright <simon@pushface.org>
Date: Tue, 08 Aug 2017 14:04:05 +0100
Subject: ANN: Cortex GNAT RTS
Newsgroups: comp.lang.ada

This release is available at Github [1] - note the move from Sourceforge.

The main motivation for the last two releases has been to support AdaCore's Certyflie [2], or at least my fork at [3].

There have been compiler interface changes, so some patching will be required [4] if you're using GNAT GPL 2016 or 2017.

New features:

- Ada.Numerics.* (except random numbers).
- Interfaces.C.Extensions.
- Ada.Real_Time.Timing_Events.
- All free store (bar a 2048-byte allowance for startup and interrupts) is available for heap allocation.
- Sequential elaboration (with the configuration pragma Partition_Elaboration_Policy (Sequential)) is supported.
- type'Image() and object'Img are supported.

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

[2] https://github.com/AdaCore/Certyflie

[3] https://github.com/simonjwright/Certyflie

[4] https://github.com/simonjwright/cortex-gnat-rts/blob/master/INSTALL.md#compatibility

[See also "Cortex GNAT Run Time Systems", AUJ 37-2, p. 72. —sparre]

AdaYaml

From: Felix Krause <contact@flyx.org>
Date: Thu, 17 Aug 2017 10:48:27 +0200
Subject: ANN: AdaYaml 0.1.0 (initial release)
Newsgroups: comp.lang.ada

I am happy to announce the first release of AdaYaml, an experimental YAML implementation in Ada 2012.

This release is made in the spirit of "release early, release often". It is still rough around the edges, features are still missing, and the API is far from stable. However, I make this release in the hope to get feedback about the general API design and usability.

The background of this library is that I am part of a working group designing YAML 1.3 and wrote this implementation to test proposed changes to YAML 1.2. This means that AdaYaml does not conform to the YAML 1.2 spec, since it already implements some of the proposed changes. However, all of these concern edge cases and I am pretty confident that a common YAML 1.2 document will parse propertly with AdaYaml.

More background and (currently sparse) documentation is available on the library's website [1], the release is available as tag of the GitHub repository [2].

[1]: https://ada.yaml.io

[2]: https://github.com/yaml/AdaYaml/tags

Ada-related Products

VectorCast for Deos

From: Vector Software
Date: Tue, 22 Aug 2017
Subject: DDC-I and Vector Software Announce Availability of VectorCAST Test Automation Platform for Deos DO-178 Safety-Critical RTOS
URL: https://www.vectorcast.com/news/vector-software-press-releases/2017/ddc-i-and-vector-software-announce-availability-vectorcast

Vector Software, the world's leading provider of innovative software solutions for embedded software quality and DDC-I, a leading supplier of software and professional services for mission- and safety-critical applications, today announced the availability of the VectorCAST test automation platform for DDC-I's Deos™ safety-critical real-time operating system (RTOS) and OpenArbor™ integrated development environment. The integrated platform greatly reduces the time and cost associated with developing, testing, and certifying DO-178 safety-critical application software.

"We're excited to be working with Vector Software to offer their world-class test automation tools for our safety-critical RTOS," said Greg Rose, vice president of marketing and product management at DDC-I. "The integration of our RTOS and tools with Vector Software's test automation suite addresses all aspects of development, test and certification for DO-178 safety-critical applications."

"We are pleased to support DDC-I's Deos RTOS with the VectorCAST test automation platform," said Jeffrey Fortin, head of product management for Vector Software. "The integration, along with both companies' DO-178 expertise, provides customers with a complete solution for a more efficient certification process."

VectorCAST is a dynamic software test solution that automates C/C++ and Ada unit and integration testing, which is necessary for validating safety- and mission-critical embedded systems. VectorCAST automates the creation of stubs and drivers as part of the creation of the test harness, normally a manual process, giving developers time to focus on building high-quality and thorough test cases. With VectorCAST, unit testing can be done natively or on a specific target or simulator. The VectorCAST Runtime Support Package (RSP) provides a full-featured integration that allows for the download, execution and results capture using the built-in networking facilities of the Deos RTOS.

Deos is a safety-critical embedded RTOS that has been certified to DO-178 DAL A since 1998. Featuring deterministic real-time response, the time- and space-partitioned RTOS employs patented slack scheduling to deliver higher CPU utilization than any other certifiable safety-critical COTS RTOS. Deos is built from the ground up for safety-critical applications, and is the only certifiable time- and space-partitioned COTS RTOS that has been created using RTCA DO-178, Level A processes from the very first day of its product development. Deos provides the easiest, lowest cost path of any COTS RTOS to DO-178 Level A certification, the highest level of safety criticality.

Development support for Deos includes DDC-I's Eclipse-based, mixed-language OpenArbor IDE, which features Ada, C and C++ optimizing compilers, a color-coded source editor, project management support, automated build utilities, and a symbolic debugger. Also included is a virtual target hardware development tool, QEMU (Quick EMUlator), which allows developers to develop, debug and test their code on their development host PC in advance of actual target hardware availability.

Ada and Operating Systems

Fedora: GNAT GPS

From: John Marino <dragonlace.cla@marino.st>
Date: Tue, 27 Jun 2017 06:36:47 -0700 (PDT)
Subject: Re: gnat-gps for Fedora 25
Newsgroups: comp.lang.ada

> Fedora 25 does not have gnat-gps available. [...]

I'm working on a cross-platform packing system.

It currently supported DragonFly BSD and Linux (FreeBSD and Solaris coming)

It contains GNAT GPS 2016 (built with GCC FSF 6.3).

It is not RPM based, but if you install it, it should work on your fedora (located at /raven/bin/gps).

see https://github.com/jrmarino/Ravenports/wiki/Ravenusers_Guide

navigate to Quickstart guide for Linux testers

This has not been announced anywhere yet, but I think it can help you in particular.

From: John Marino <dragonlace.cla@marino.st>
Date: Tue, 27 Jun 2017 06:40:18 -0700 (PDT)
Subject: Re: gnat-gps for Fedora 25
Newsgroups: comp.lang.ada

> [...]

FYI, step 4 you'd do something like "sudo /raven/sbin/pkg install gps-complete-standard" to install the prebuilt package.

List of current linux packages: http://muscles.dragonflybsd.org/misc/Ravenports/Linux%3A2.6.32%3Aamd64/All/

Mac OS X: GNAT GPL for ARM-EABI

From: Simon Wright <simon@pushface.org>
Date: Sat, 1 Jul 2017 16:51:47 +0100
Subject: GNAT GPL 2017 for arm-eabi on macOS
Newsgroups: gmane.comp.lang.ada.macosx

This is GNAT GPL 2017, rebuilt as a cross-compiler from Mac OS X to arm-eabi. The CPUs supported include cortex-m3, cortex-m4, cortex-r4.

The runtimes from the AdaCore gnat-gpl-2017-arm-elf-linux-bin are included:

- ravenscar-full-rpi2
- ravenscar-full-stm32f4
- ravenscar-full-stm32f429disco
- ravenscar-full-stm32f469disco
- ravenscar-full-stm32f746disco
- ravenscar-full-stm32f769disco
- ravenscar-full-tms570
- ravenscar-full-zynq7000
- ravenscar-sfp-rpi2
- ravenscar-sfp-stm32f4
- ravenscar-sfp-stm32f429disco
- ravenscar-sfp-stm32f469disco
- ravenscar-sfp-stm32f746disco
- ravenscar-sfp-stm32f769disco
- ravenscar-sfp-tms570
- ravenscar-sfp-zynq7000
- zfp-lm3s
- zfp-rpi2
- zfp-stm32f4
- zfp-stm32f429disco
- zfp-stm32f469disco
- zfp-stm32f746disco
- zfp-stm32f769disco
- zfp-tms570
- zfp-zynq7000

as are the examples in share/examples/gnat-cross/.

The compiler is known to run on Sierra and El Capitan.

Find at https://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2017-arm-eabi-darwin-bin/

[See also “Mac OS X: GNAT GPL for ARM-EABI”, AUJ 37-2, p. 78. —sparre]

Debian: GNAT

From: Nicolas Boulenguez <nicolas.boulenguez@free.fr>
Date: Fri, 28 Jul 2017 14:59:02 +0200
Subject: gcc-7 migration
Newsgroups: gmane.linux.debian.packages.ada

In testing and unstable, most Ada packages FTBFS because some changes libgnat6-dev invalidate their .ali files. Fixing this requires a source upload and a passage through the NEW queue for every library.

[1] https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=866355

In experimental, the migration to gnat-7 is on its way. Most FTBFS are also caused by obsolete .ali files, but it is sufficient to recompile as the ALI/SO versions are already diffenent *from the ones in unstable*.

asis, aunit, dh-ada-library, florist, gmpada, gnatcoll, gpr, gtkada, ncursesada, opentoken, templates-parser, texttools, xmlada and xmlezout are mostly completed.

adabrowse, adacgi, adacontrol, adasockets, ahven, alog, anet, aws, dbusada, gnat-gps, liblog4ada, pcscada, polyorb, spark and topal need to be updated.

Now that gcc-7 is available in testing, and its ali files do not change anymore, I suggest that we start uploading to unstable.

From: Nicolas Boulenguez <nicolas.boulenguez@free.fr>
Date: Sat, 12 Aug 2017 19:52:26 +0200
Subject: gcc-7 migration
Newsgroups: gmane.linux.debian.packages.ada

The gnat package now depends on gnat-7, meaning that gnat-7 is the default Ada compiler in the unstable distribution.

Each source package building an Ada library requires a source+binary upload with new ALI and SO versions (source-only uploads are not allowed in the NEW queue).

Each source package depending on an Ada library requires a source upload with an updated -dev build-dependency, once the dependency has passed through the NEW queue.

Feel free to contact me for advice, review and/or upload sponsorship.

Ubuntu: Simple Components etc.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 11 Aug 2017 15:19:19 +0200
Subject: ANN: Ubuntu 17.10 packages available
Newsgroups: comp.lang.ada

Ubuntu 17.10 packages of

- Ada industrial control widget library
- Fuzzy machine learning framework
- Fuzzy sets for Ada
- GtkAda 3.14.2
- GtkAda contributions
- MAX! home automation
- Interval arithmetic for Ada
- Units of measurement for Ada
- Simple components for Ada
- String edit
- Tables

are available at www.dmitry-kazakov.de. The packages can be downloaded individually or via repository. The repository can be added by placing the following line into /etc/apt/sources.list:

deb [trusted=yes] http://www.dmitry-kazakov.de/distributions/ubuntu zesty main

References to Publications

Automating Test Generation

From: Rapita Systems
Date: Thu, 20 Jul 2017
Subject: Automating test generation with AUTOSAC
URL: https://www.rapitasystems.com/blog/automating-test-generation-autosac

As anyone working with safety-critical software knows, testing is a costly process. DO-178 and ISO 26262 guidelines set standards for software quality assurance that require significant effort to meet. In the avionics software industry, it has been estimated that over 50% of the cost of overall development is spent on testing!

How much of this cost could be avoided if test requirements were written in a computer-parseable language so that tests can be generated automatically from them?

In January 2016 we partnered with Altran UK, Rolls-Royce and the Universty of Oxford to answer just that. In the 15-month NATEP-funded AUTOSAC project, we aimed to AUTOmate the generation of tests from Spark Ada Contracts.

[...]

Ada Inside

MAX! Home Automation

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 26 Jul 2017 15:43:59 +0200
Subject: ANN: MAX! home automation v1.10
Newsgroups: comp.lang.ada

MAX! home automation is a GTK+ application to manage ELV/eQ-3 MAX! cubes. A cube is a gateway to a network of radiator thermostats, shutter contacts etc.

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

Changes to the previous version:

- HTTP requests added to get the average, maximum, minimum valve positions;
- MQTT topics added to subscribe to the average, maximum, minimum valve positions.

[See also “MAX! Home Automation”, AUJ 38-2, p. 78. —sparre]

SparCanto

From: Ken O. Burtch <koburtch@gmail.com>
Date: Thu, 17 Aug 2017 20:17:18 -0700 (PDT)
Subject: ANN: SparCanto Prototype
Newsgroups: comp.lang.ada

SparCanto is a web framework and CMS written in SparForte, my Ada-based scripting language. It is for development of SparForte web applications.

This version is a proof-of-concept and not yet ready for general use. I had to temporarily suspend work on SparCanto due to other projects.

Contributors that want to work on SparCanto can access it on GitHub. There is a link to it on the SparForte website download page.

Pasta!

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue, 29 Aug 2017 17:49:39 +0000
Subject: Pasta! with High Scores
Newsgroups: gmane.comp.lang.ada.gnoga

From now on there is a high score list appearing at the end of each successfully completed level of “Pasta!” [1].

Actually the best scores were already recorded for the last couple of days, but not displayed.

[1] http://pasta.phyrama.com/game.html

[See also “Pasta!”, AUJ 38-2, p. 78. —sparre]

Job

From: Real-Time Innovations
Date: Thu, 31 Aug 2017
Subject: Senior Ada Software Engineer Industrial IoT at Real-Time Innovations
URL: https://boards.greenhouse.io/rti/jobs/781054#.WaXvpNOGPwO

As a senior Ada software engineer in our development team, you will be part of a team of experts building a secure real-time middleware platform with extreme performance and scalability. Focussed on our Ada language binding, your work will directly impact RTI customers, e.g., building reliable radar systems and avionics applications.

The RTI Connext software enables 100s and 1000s of applications and devices to exchange data in a timely and reliable way. Our software features direct peer-to-peer connections, reliable multicast, automated application discovery, and unique, contractual quality-of-service control. Our team values creativity, risk-taking, innovation, and open communication.

So, what do we do? Simply put, RTI connects smarts to distributed systems. We seek to transform entire industries. Our hottest markets are today’s most exciting: autonomous cars, smart medical systems, green energy, unmanned planes. Our software smartly runs the largest power plants on the continent, connects perception to control in vehicles, drives the new generation of medical robotics, controls hyperloop and flying cars, and provides 24x7 medical intelligence to hospital patients and emergency victims. We are making the world greener, safer, faster, and flat-out cooler.

[...]

Requirements

- Experienced programmer. Hands-on experience with the Ada programming language. You love programming languages. You will also work in other programming languages. We support C/C++/Java/C#/Python/Javascript/Lua.
- Degree in Computer Science or related field (Advanced degree preferred). Studies related to distributed systems, peer-to-peer networks, and computer networking
- Ability to work successfully in a highly distributed team, including headquarters in USA.
- Excellent English written communication skills, with the drive and desire to improve English verbal communication skills

[...]

Ada in Context

Compile-time Checking of Access Types?

From: Nick P. <digitalkevlar@gmail.com>
Date: Sat, 13 May 2017 13:33:27 -0700 (PDT)
Subject: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

I'm a high-assurance engineer/researcher who mainly evangelizes the use of proven methods, tools, etc in proprietary and FOSS systems. I've promoted Ada long time (esp Barnes' Safe and Secure book) but I don't use it myself so I need help answering something. Inspired by Cyclone language and linear types, the Rust language has pulled off a rare feat in using the ownership and type system (esp affine types) to eliminate temporal errors that come from mismanaging references. That plus two "traits" eliminates race conditions, too. This is not in static code or something with heavy restrictions on what the code can do. Their thriving community is coding about everything you can think of from desktop (eg Redox OS) to servers (eg TrustDNS) to embedded (eg Tock OS). Here's a description of their memory-safety scheme:

https://doc.rust-lang.org/book/references-and-borrowing.html

Here's the Cyclone page for people just curious where it came from or on safe languages in general:

https://en.wikipedia.org/wiki/Cyclone_(programming_language)

So, with Rust's approach, they get memory safety even for *dynamic or concurrent use* of memory at compile time with no overhead, runtime checks, GC, etc. Whereas, the last thing I read on Ada showed it has a few tricks but many dynamic uses resort to unsafe deallocations at some point. Other people were suggesting reference counting or a GC leading me to further think it lacks this ability of Rust. So, my question is, does Ada 2012 currently have Rust's capability to enforce both temporal, memory safety and immunity to race conditions? I'm really focusing on an equivalent to the borrow-checker in Rust, though. If it doesn't have an equivalent, is there anyone working on adding it to Ada esp at AdaCore? What Ada/SPARK have already + memory safety for dynamic code would be an awesome combination that would put Rust in distant second. [...]

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sat, 13 May 2017 23:19:26 +0200
Subject: Re: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

> [...]

This looks sort of like Ada's accessibility levels and accessibility rules, from ARM 3.10.2, though as it says there, "In most cases, accessibility is enforced at compile time by Legality Rules. Run-time accessibility checks are also used, since the Legality Rules do not cover certain cases involving access parameters and generic packages."

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Sun, 14 May 2017 13:19:53 +0300
Subject: Re: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

> [...]

I agree that Ada accessibility rules are related to Rust's scoped lifetimes, but my impression (after a brief read of the Rust "borrow-checker" material) is that the Rust scheme goes a lot further than what is today standard in Ada. For example, AIUI Rust makes it impossible to try to dereference a null pointer, and Rust also completely prevents dangling references, even when dynamically allocated objects are deallocated.

In a multi-threaded program, again AIUI, Rust statically prevents concurrent writes from different threads to the same variable. That is "legal" in Ada, but (as discussed in a concurrent thread on "Portable memory barriers") Ada has unchecked rules on when such access is non-erroneous.

AIUI the Rust scheme is based on (a) compile-time tracking of the set of references that refer to a given object, as well as the kind of access (read-only, or read-and-write) that each reference allows, and (b) wrapping all possibly-null references into "Optional" types (similar to Ada's variant records) to hide the "null" values.

It is not clear to me if these Rust advantages bring with them some restrictions on the kinds of data structures that a Rust program can use, or require some Rust-specific idioms for transforming traditional reference-heavy data structures (for example graphs) into Rust form.

I hope "someone" will make an in-depth study of how the advantages of the Rust scheme could be imported into Ada. I'm afraid it may be rather hard to do, as Rust references are so different from Ada's access values.

From: Nick P. <digitalkevlar@gmail.com>
Date: Sun, 14 May 2017 09:46:04 -0700 (PDT)
Subject: Re: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

> [...] most programmers consider code that needs run-time checking as a bug

I can see why they'd avoid such constructions if their tooling couldn't prove them safe. If another tool can, though, then I'm thinking it's a bug in Ada in that Ada's model or tools can't handle that analysis at compile-time. Something worth fixing with R&D. That's why I'm trying to assess what Ada can do currently in this area. Several other languages knock this problem out at compile time. They're functional, logical, and imperative. So, I know it is feasible in general case.

Niklas Holsti's post shows he understands what capability I'm describing. Rust code can be shown free of double-free's and dangling-pointers at *compile-time* with *no runtime checks or GC*. It does it with just a few simple rules. Here's the simplest, shortest description I could find to save you all time:

https://stackoverflow.com/questions/36136201/how-does-rust-guarantee-memory-safety-and-prevent-segfaults

Those are combined with "traits"... or something else in language... to allow race-free concurrency. Instead of mandating one model for language, various models of concurrency are defined in libraries to let you use model easiest for your problem. Language's ownership model & borrow-checker ensures they're all memory-safe and race-free along with any code using them. So, you get these guarantees even in multi-threaded code doing many allocations and de-allocations of memory dynamically. New developers do have a hard time fighting with the borrow-checker early on. My meta-analysis of their comments indicates much of it is intrinsic to safely handling ownership and borrowing of references that C and GC'd languages didn't teach them. The tool just enforces rules (i.e. affine types) known to work. Learning curve is about 1-2 months of practice until they say borrow-checker rarely has problems with their code.

So, can someone today use Ada in a straight-forward way to write single- or multi-threaded applications that are, in every use-case, totally immune at compile-time to use-after-free and double-free errors with zero, runtime overhead? Or can it not do that?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 14 May 2017 19:18:45 +0200
Subject: Re: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

> [...]

It would be helpful to have an example where pointers are needed. The example provided requires no pointers. So in Ada case, no such problem exist at all.

> [...]

Without having other examples, the answer is: there is no such problem in Ada because arguments of task rendezvous are not pointers.

For multitasking, problems arise when the method of problem decomposition requires constructs which are not safe in the sense that safety is statically undecidable. Which means that a decidable static constraints would simply kill the algorithm.

Considering the problem of having tasking safe per construction, my impression is that constraints are no or very little help. Additional methods of decomposition are.

From: Yannick Moy <moy@adacore.com>
Date: Sun, 14 May 2017 14:28:34 -0700 (PDT)
Subject: Re: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

In fact, we currently have at AdaCore an intern working with us on the inclusion of Rust-like pointers in SPARK. He has reached a first milestone which was the description of suitable rules to include safe pointers in SPARK, which have convinced the SPARK Language Design Group at AdaCore and Altran UK (the small group working on the evolutions of the SPARK language).

He's now working with us and researchers from Inria team Toccata to give a mathematical semantics to the notions that we're using for these safe pointers: move (on assignment mostly), borrow (on parameter passing for mutable objects) and observe (on parameter passing for immutable objects). We have also started looking at the concrete implementation of these rules in GNATprove (the SPARK analysis tool).

In this work, we don't target everything that the Rust borrow checker does:

- we leave accessibility checking (the lifetime checking in Rust) to the compiler, using existing Ada rules, plus some restrictions in SPARK to avoid the need for dynamic accessibility checks
- we leave nullity checking to proof (a Verification Condition will be generated for dereference of possibly null pointers), with the help here of Ada non-null types that reduce the need for such proofs. Given that pointers are always initialized to null in Ada, there is no need to separately deal with initialization.
- we ignore the problem of memory leaks (which could be tackled later as an extension of the current scheme)

So the main issue that we really address with this work is the issue of non-aliasing. Or rather the issue of problematic interferences, when two names, one of which can be updated, are referring to the same memory location. We're focusing on this issue, because it is the one preventing inclusion of pointers in SPARK, as for formal analysis we rely on the ability to perform modular analysis, where we make assumptions on the absence of problematic interferences.

But since our solution to non-aliasing is based on this Rust-like notion of ownership of pointers, the same solution will also forbid use-after-free or double-free.

This work is ongoing, we will certainly let the community know about our progress after the summer.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Sun, 14 May 2017 22:59:55 +0300
Subject: Re: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

[...]

> It's very rare for well designed Ada to need access types.

"Well designed" is of course subjective. The container library has made it practical to avoid access types in the application code, but then there are other potential run-time problems, such as "tampering" with the containers, which require run-time checks (and which are to some extent consequences of the use of access types within the container library).

> An overwhelming majority of applications can be implemented without ever writing "access".

I find it difficult to agree with that "overwhelming", at least if one includes the access types used under the covers in the container library.

Even in applications where heap allocation is forbidden, there are usually some dynamically allocated resources -- elements of "resource pools" such as message buffers -- with the corresponding application-defined "reference" data types, and the same problems of managing allocations over time. I don't know if Rust's memory-management scheme extends to such non-heap "references, however.

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Mon, 15 May 2017 18:23:07 +0200
Subject: Re: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

> [...]

I'm pretty sure the "tampering" restrictions in the containers have nothing to do with possible implementations (which need not even be in Ada), and everything to do with maintaining the integrity of the structures. They're intended to ensure that an ordered container doesn't have an element out of order, or a hashed container, one with a different hash than its bin.

> [...]

There's nothing about using the containers that requires the user to write "access", so clearly they should not be included.

One might want to use 'access to pass a subprogram as an anonymous access-to-subprogram parameter of a container operation, but since such things can't be assigned and can't be freed, they're not really access types, but rather a strange syntax for limited subprogram types.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 15 May 2017 18:19:43 -0500
Subject: Re: Rust's temporal safety for Ada/SPARK
Newsgroups: comp.lang.ada

> [...] It's very rare for well designed Ada to need access types. [...]

I would suggest that Jeff's answer here should be read to say that "well-designed Ada code has no need for dynamic checks". Ada has plenty of unsafe constructs (for low-level memory managment, machine access, and the like), but one never has to use them. Ada after all is about combining safety with capability -- we try to allow everything (including unsafe stuff), but try to make it clear what is unsafe so that code review tools can determine what issues exist.

In any case, Ada code can be written so that there are no dereference checks and no dangling pointers. You obviously lose some capability when one does so. Does that matter? Depends on what you are doing.

Static rules (of any sort) are always going to reduce capability, *especially* when it comes to multithreaded programs. I find it highly unlikely that any sort of static rules would actually work in "every use-case". That's been claimed many times in the past, and it has always turned out that the claims were *way* overblown. The people making such claims often don't understand multithreading well enough. (It's an area, like random numbers and probability, that a little knowledge is more dangerous than no knowledge.)

I'm also very suspicious of any claims of new static rules simply because OOP pretty much forces dynamic checks if one uses references; strong typing breaks down for that and everything essentially becomes dynamic. There probably aren't any dereference checks, but you end up with dynamic type checks instead (substantially worse). Best thing is to avoid references altogether (but of course that too reduces capability).

Object_Size Attribute

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 7 Jun 2017 19:51:38 -0500
Subject: Re: Arrays in Ada 2012
Newsgroups: comp.lang.ada

> I always use rounding in the corresponding cases
>
> (Long_Long_Integer'Size + Storage_Unit - 1) / Storage_Unit;
>
> Is this an overkill from the point of view the permissions RM gives to array implementation?

The definition of 'Size is stupid, so you need rounding like that to be portable. It's probably not necessary for base types, but if you had

(Natural'Size / Storage_Unit)

you would get the wrong answer (Ada requires Natural'Size to be one less than Integer'Size, so it usually is 15 or 31 or 63).

Probably a better solution is to use 'Object_Size for such expressions, but that is only portable to Ada 202x compilers (for which none exist yet for obvious reasons). Object_Size usually will be a multiple of the storage unit; that's the Implementation Advice for it, but of course some implementation could ignore that in some circumstance.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 8 Jun 2017 09:07:41 +0200
Subject: Re: Arrays in Ada 2012
Newsgroups: comp.lang.ada

> The definition of 'Size is stupid, [...]

You mean being bit size or being kind of weakly typed (universal integer). I disagree on the first and agree on the second. It possibly should be overloaded:

```
function 'Size (...)
  return Storage_Count;-- Storage units
function 'Size (...)
  return Storage_Size;  -- Bits
```

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 8 Jun 2017 22:23:26 -0500
Subject: Re: Arrays in Ada 2012
Newsgroups: comp.lang.ada

> [...]

The definition of 'Size is stupid, because it's neither the bit size nor the usual allocation size, but a weird hybrid of both. It's primary effect is to provide a lower bound for packing, which is hardly ever what you want to limit. (Why would you want to prevent someone from packing some component?)

And 'Size has no effect at all on what you can write in other rep. clauses, like 'Component_Size and record representations. So setting 'Size almost never does what you want (unless the compiler tries to be friendly -- but that runs into problems with the required default value of Size).

Thus GNAT (and soon Ada) introduced 'Object_Size, which gives the UPPER bound on the allocated size. A much more useful thing to bound - you can ensure that type Byte really only uses a byte, for instance.

Array Aggregates

From: Ivan Levashev <bu_gen@octagram.name>
Date: Sun, 18 Jun 2017 09:14:12 +0700
Subject: Re: Arrays in Ada 2020
Newsgroups: comp.lang.ada

FYI Ada 2020 is about to bring some improvements:

http://www.ada-auth.org/standards/2xrm/html/ RM-4-3-3.html#p45

```
> G : constant Matrix :=
    (for I in 1 .. 4 =>
      (for J in 1 .. 4 =>
        (if I = J then 1.0 else 0.0)));
        -- Identity matrix
```

Proposal: Maximum_Size Aspect

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Fri, 23 Jun 2017 22:21:33 +0300
Subject: Re: Ada Annoyances
Newsgroups: comp.lang.ada

> [...]

Sometimes usage rules are imposed by environment constraints, in particular limited resources in smallish embedded systems, combined with reliability requirements which mean that running out of resources at run time must be avoided.

> [...]

I have so far avoided using tagged types in my embedded applications because they indeed hamper the discovery of resource usage (execution time and stack space) by static analysis, as you said.

There are two reasons why tagged types hamper such analysis:

a) dispatching calls (as you said), where the actual callee is determined by run-time values (tags) which are hard to predict by static analysis

b) the non-static size of class-wide objects (of type T'Class), which means that the compiler and/or the programmer must use dynamic allocation (usually heap or secondary stack) for such objects.

Point (a) can be worked around: static analysis tools usually let the analyst specify the possible set of callees for a "dynamic call" (of which dispatching calls are one kind) and the analysis can then encompass all those callees. (Alternatively, the analysis tool can extract the class hierarchy from the debugging information, and itself discover the possible callees.)

Point (b) is more difficult and I know of no work-around that can be applied at analysis time.

For some time, I have had in mind a possible Ada extension to solve point (b): an attribute/aspect that would let the programmer set a static upper bound on the size of any object in T'Class. If we call this aspect Maximum_Size (or perhaps Maximum_Size'Class), the programmer could use it like this:

```
type Root is
  tagged record ... end record;
with Maximum_Size => 128;
type Child is new Root
  with record ... end record;
-- The compiler checks that Child'Size is
-- at most 128 bits, and
-- rejects the program otherwise.
```

It would now be legal to create statically sized data structures using Root'Class, without dynamic memory allocation, by allocating 128 bits for each value of type Root'Class:

```
type Object_List is
  array (List_Index) of Root'Class;
type Object_Pair is record
  A, B : Root'Class;
end record;
```

and so on.

With this extension, or some other means to solve point (b), I would start using tagged types in embedded SW. For example, I have a major SW component, used in several projects, which simulates a class hierarchy with variant records and case statements. This component would be greatly improved by using a tagged type instead, but it would need data structures with class-wide components of static (maximum) size.

What do people think of a Maximum_Size aspect? Should I consider writing a formal suggestion to ada-comment?

[See also “Community Input for the ARG” earlier in this issue. —sparre]

From: Edward R. Fish <onewingedshark@gmail.com>
Date: Fri, 23 Jun 2017 21:02:59 -0700 (PDT)
Subject: Re: Ada Annoyances
Newsgroups: comp.lang.ada

> [...] Should I consider writing a formal suggestion to ada-comment?

Please do.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Sat, 24 Jun 2017 23:56:58 +0300
Subject: Re: Ada Annoyances
Newsgroups: comp.lang.ada

> GNAT is happy with

> **type** Parent **is tagged null record**

> **with** Dynamic_Predicate => Size (Parent) < 128;

> **function** Size (P : Parent'Class) **return** Integer **is** (P'Size);

> **type** Large **is array** (1 .. 10) **of** Integer;

> **type** Child **is new** Parent **with record**

> L : Large;

> **end record**;

>

> Declaring an object of type Child raises Assert_Failure.

As one would expect, based on standard Ada, yes?

> Of course you'd much rather have a static compile-time check!

Indeed I would.

But the check is not the main point in the suggested Maximum_Size aspect: the main point is that it would let the compiler consider the type Root'Class as a definite subtype, and would therefore allow its direct use as a component of arrays or records, instead of forcing an access-classwide to be used as an intermediate.

I don't suppose GNAT lets you use Parent'Class as the component type of an array, even with this Dynamic_Predicate?

There may however be some other semantic implications of the definite vs indefinite subtype divide, not related to the size of the values, that would make it hard to let the suggested Maximum_Size aspect change the classwide type from indefinite to definite.

From: Niklas Holsti
<niklas.holsti@tidorum.fi>
Date: Mon, 26 Jun 2017 23:20:01 +0300
Subject: Re: Ada Annoyances
Newsgroups: comp.lang.ada

>> Point (b) [...]

> Simply banning the use of T'Class has that effect.

At design and compilation time, yes, with consequent restrictions on the design. But it is not a work-around that allows static analysis of programs which do use T'Class.

> It's rather drastic, but it eliminates all of the dynamic features. Note that this was considered important enough that the standard (in Annex H) restriction No_Dispatch has this effect.

> You still get the other advantages of tagged types (extension, proper inheritance for private, equality, prefixed notation, etc.), and there is almost no runtime penalty (or analysis problem).

For the application I most have in mind (a SW component currently using discriminated records to simulate tagged types) there would be no point in using tagged types with the No_Dispatch restriction. The loss of class-wide programming and class-wide data structures would remove almost all benefits.

My hope is that a Maximum_Size aspect would let one manipulate class-wide objects in the same "definite" way as is possible for variant records with a default discriminant value. However, it would require that the tag of an object could be changed by assignment; this is perhaps too radical a change in the tagged object semantics.

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Mon, 26 Jun 2017 16:47:26 -0500
Subject: Re: Ada Annoyances
Newsgroups: comp.lang.ada

> [...]

There's no free lunch! T'Class is by definition "indefinite", since existing code already compiled has to be able to handle newly defined types (including those that don't yet exist). That's never going to allow conventional static analysis.

> [...] it would require that the tag of an object could be changed by assignment; [...]

That seems way too drastic a change. I could imagine an aspect like the one you proposed to get rid of the indirection, but changing the semantics in a major way seems to be more than aspects are supposed to do. And changing the tag via assignment means having to be prepared to change finalization of objects after the fact as well. Ugh.

From: Robert A Duff
<bobduff@TheWorld.com>
Date: Thu, 29 Jun 2017 17:12:20 -0400
Subject: Re: Ada Annoyances
Newsgroups: comp.lang.ada

> There are two reasons why tagged types hamper such analysis:

As Randy pointed out, it is more correct to say that class-wide types do that. Tagged types by themselves do not cause these problems. Tagged types without class-wide types are not super useful, but they are somewhat useful.

> **type** Root **is tagged record** ... **end record**
with Maximum_Size => 128;

Something like that was considered and rejected for Ada 9X. Part of the problem is that it seems so low level to be talking about sizes. It's not even portable. And not maintainable -- if you delete a big type, or make it smaller, you're now wasting space.

It would be better to have the compiler compute the maximum size needed. That would require the compiler to do a fairly global analysis, which is something Ada compilers are not set up to do.

> **type** Object_List **is array** (List_Index) **of** Root'Class;

> **type** Object_Pair **is record**

>   A, B : Root'Class;

> **end record**;

Would you allow:

X : Object_List (1..10);

Y : Object_Pair;

? If so, what is the 'Tag of the various components? "Undefined" is not a very satisfying answer.

These things are analogous to records with defaulted discriminants. The language makes some (unsuccessful!) attempt to prevent uninitialized discriminants.

From: Niklas Holsti
<niklas.holsti@tidorum.fi>
Date: Tue, 4 Jul 2017 22:30:23 +0300
Subject: Re: Ada Annoyances
Newsgroups: comp.lang.ada

> [...] Part of the problem is that it seems so low level to be talking about sizes. It's not even portable.

You are right, but this feature would be used only or mainly in embedded programs, which usually already contain unportable, low-level stuff: task stack sizes, record representation clauses, etc.

> And not maintainable -- if you delete a big type, or make it smaller, you're now wasting space.

If the feature would be adopted, I imagine a friendly compiler would (at least optionally) tell me that I am wasting space, or that the given Maximum_Size is too small, and what would be the minimum valid value (as GNAT does now for 'Size clauses that give a too-small size) even if it requires some bind-time or link-time global check. That is a quality-of-implementation issue.

> It would be better to have the compiler compute the maximum size needed. [...]

Well, the "binder" part of the compilation system does some global stuff. And I would not be surprised if link-time "relocation"-type computations could be (mis-)used to compute the maximum size of any type in a class.

Some compilers already support stack-size analysis, which is a similar global analysis.

> [...]

I would like to allow that (default initialized components of type Root'Class), but I would not much mind having to initialize such components explicitly.

> If so, what is the 'Tag of the various components? "Undefined" is not a very satisfying answer.

I agree that "undefined" would not be good. The natural answer seems to be that the default initial tag is that of the Root type, but then we must assume that it is not an abstract type.

> These things are analogous to records with defaulted discriminants.

Yes, that has been my mental model (and it is the implementation used in the main SW component that I would like to switch over to class-wide types).

> The language makes some (unsuccessful!) attempt to prevent uninitialized discriminants.

Interesting -- I did not know that the attempt is not fully successful. Is it easy to explain when the attemp fails?

From: Robert A Duff
<bobduff@TheWorld.com>
Date: Wed, 05 Jul 2017 16:03:47 -0400
Subject: Re: Ada Annoyances
Newsgroups: comp.lang.ada

> [...]

I would use it even in nonembedded systems.

I don't buy the idea that just because some of one's embedded code needs to be nonportable/low-level, it's OK to force other stuff to be, when it's not logically necessary.

If you're not interfacing with external hardware or similar, the compiler should compute record layouts/sizes.

> [...]

If it can warn about too-large max size, then it can compute the max size for you. Yes, it has to be a global analysis.

Too-small max size is easy -- that can be done at compile time for each type.

> [...]

Well, if you implemented this feature in GNAT, you'd be proven right. I don't think AdaCore is going to spend time on it any time soon.

> [...] global analysis.

Yes. ARG has always shied away from requiring such things.

> I would like to allow that (default initialized components of type Root'Class), but I would not much mind having to initialize such components explicitly.

OK.

> [...] The natural answer seems to be that the default initial tag is that of the Root type, [...]

But root types are usually abstract.

> Is it easy to explain when the attemp fails?

Quite easy:

```
Uninit : Integer;
X : Some_Record (Discrim => Uninit);
```

In a simple case like that, compilers likely warn. But it's not hard to hide the uninitialized variable from the compiler. E.g. a component of a heap-allocated record -- compilers can't warn about that, because it would cause too many false positives.

Strings and Text

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 29 Jun 2017 19:23:02 +0200
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> The current string mess. Remove it all and start again.

While at removing the string mess, remove strings altogether!

Strings' use cases are almost always just UI related: be it an exception message, compiler diagnostic, an alert box, and so forth: in the end, strings are almost always at the human interface level. (Note that hashed keys need not be strings.) UI in just strings?

The presence of bare strings in programs is only a legacy and reminiscent of

- lack of typed data,
- our habits that move towards ADTs slowly.

E.g., a notion of Message conveys not just an array of characters. Microformats of text exist only because lenient management is tolerant of stuff that "tends to work in 90% of the cases", and "everyone else does it like this, plus we're not disruptors". IT business could use that tolerance of bare strings in more meaningful IT situations, those that promise higher ROI than messing about strings.

Note that this posting is demonstrably not just a string.

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Thu, 29 Jun 2017 20:27:07 +0200
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> While at removing the string mess, remove strings altogether!

Interesting point. While I think I can follow you, I still think we would need to have quite a lot of string-like types in the language anyway (file names, directory names, host names, user-entered characters, user-readable characters, XML, HTML, Ada identifiers, Ada comments, Ada source file lines, ...).

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 30 Jun 2017 07:27:07 +0200
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> There's just so much stuff I need unicode strings for and we just don't have good enough support for it.

Well, you have Wide_String if you can stay within the BMP, or Wide_Wide_String if you need the whole Unicode. There are packages for character characterization/translation. You have packages for encoding/decoding UTF8/16/32. What is the extra support you need?

From: Luke A. Guest <laguest@archeia.com>
Date: Fri, 30 Jun 2017 04:23:31 -0700 (PDT)
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> 1) Character database.

> 2) Iterators over code points, word boundaries, grapheme clusters, etc.

> 3) BIDI iterators, if your want to render internationalised text.

> That's just for starters. Like I said, a full implementation which is in the standard, not some half arsed thing which is scattered all over the place.

You just reminded me...

4) Normalisation conversion.

5) Sorting.

6) Unicode regular expressions.

7) Streaming.

8) Unbounded Unicode strings.

The amount of text processing people need to do in the 21st century is massive, Ada should make this easy to do, but it doesn't. Ada needs it's arse dragging into the modern world.

Unicode in File Names (and other places)

From: Simon Wright <simon@pushface.org>
Date: Tue, 04 Jul 2017 14:57:03 +0100
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> [...] GNAT smashes the file name to lower case if it knows that the file system is case-insensitive (using an ASCII to-lower, so of course “smash” is the right word if there are UTF-8 characters in there). [...]

It's worse than that, on MacOS anyway [2].

```
$ GNAT_FILE_NAME_CASE_
SENSITIVE=1 gnatmake -c p*.ads
gcc -c páck3.ads
páck3.ads:1:10: warning: file name does not match unit name, should be "páck3.ads"
```

The reason for this apparently-bizarre message is [3] that MacOS takes the composed form (lowercase a acute) and converts it under the hood to what HFS+ insists on, the fully decomposed form (lowercase a, combining acute); thus the names are actually different even though they _look_ the same.

I have to say that, great as it would be to have this fixed, the changes required would be extensive, and I can’t see that anyone would think it worth the trouble.

The recommendation would be “don’t use international characters in the names of library units”.

[2] https://gcc.gnu.org/bugzilla/show_bug.cgi?id=81114#c1

[3] https://stackoverflow.com/a/6153713/40851

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 5 Jul 2017 07:25:11 +0200
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

[...]

> One of unicode's biggest problems is that there's no longer any coherent vision -- it started off as a idea to offer one code-point per character in human language, but then shifted to glyph-building (hence combining characters), and as such lacks a unifying principle.

The unifying principle is the normalization forms. The fact that there are several normalization forms comes from the difference between human and computer needs.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 5 Jul 2017 07:21:31 +0200

Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> [...] thus the names are actually different even though they _look_ the same.

Apparently, they use NFD (Normalization Form D). Normalization forms are necessary to avoid a whole lot of problems, although Ada requires normalization form C (ARM 2.1 (4.1/3)), or more precisely, it is implementation defined if the text is not in NFC.

From: Simon Wright <simon@pushface.org>
Date: Wed, 05 Jul 2017 10:47:39 +0100
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> [...] implementation defined if the text is not in NFC.

That reference specifies NFKC which I suppose is near! GNAT uses this if either you compile with -gnatW8 or the file begins with a UTF8 BOM.

The problems I've noted in this thread in the GNAT implementation are two:

(1) On Windows and MacOS (and possibly on VMS, not sure if that's relevant any more) the file name corresponding to a unit name is converted to lower-case assuming it's Latin-1 - System.Case_Util.To_Lower,

```
function To_Lower (A : Character)
     return Character is
   A_Val : constant Natural :=
     Character'Pos (A);
begin
   if A in 'A' .. 'Z'
     or else A_Val in 16#C0# .. 16#D6#
     or else A_Val in 16#D8# .. 16#DE#
   then
      return Character'Val (A_Val + 16#20#);
   else
      return A;
   end if;
end To_Lower;
```

This is the problem that prevents use of extended characters in unit names.

(2) On MacOS, the expected file name appears to be stored in NFC, but is retrieved from the file system in NFD.

It seems this will only cause a problem if you compile the file (on its own, not as part of the closure of another file - weird - possibly because the wildcard picks up the NFD representation, while compiling as part of the closure uses the NFC representation in the ALI?) with -gnatwe:

```
$ GNAT_FILE_NAME_CASE_
SENSITIVE=1 gnatmake -c -f p*.ads
-gnatwe

gcc -c -gnatwe páck3.ads

páck3.ads:1:10: warning: file name does
not match unit name, should be
"páck3.ads"

gnatmake: "páck3.ads" compilation error
```

[I'm unable to replicate the difference between á and á here. --sparre]

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 5 Jul 2017 13:20:53 +0200
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> That reference specifies NFKC which I suppose is near!

Not that near when it comes to ligatures and other crazy characters... But you are right, it's NFKC.

> GNAT uses this if either you compile with -gnatW8 or the file begins with a UTF8 BOM.

Actually, this has nothing to do with encoding or coded character sets. Even if you use Latin-1, the set of allowed characters is defined as those that belong to NFKC.

> [...]

I can talk about character issues since I gave that tutorial at AE'17...

How operating systems manage that, I don't know.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 5 Jul 2017 13:42:14 -0500
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> But you are right, it's NFKC.

Actually, you were right the first time, but it doesn't show up in the Ada 2012 as this is a recent correction (recall AI12-0004-1? It was just approved by WG 9 at the June meeting). NFKC is *definitely* the wrong rule.

Note that we chose NFC in part because WC3 recommends that all Internet content be in NFC, and because it is the more compact representation. I'm surprised that anyone would use NFD (since it can be three times larger than NFC), but I suppose I shouldn't ever be surprised by the choices of others. ;-)

As always, you can see the *current* state of Ada by using the working draft RM (see http://www.ada-auth.org/standards/ada2x.html). For this rule, that is 2.1(4.1/5). [http://www.ada-auth.org/standards/2xrm/html/RM-2-1.html#p4.1]

I suppose the working draft is a bit confusing for this use (that is, Ada-Comment) as corrections (like this) take effect immediately upon WG 9 approval while amendments don't take effect until the next Standard update. You can tell them apart by looking at the bottom of each subclause at the "<something> from Ada 2012" (for instance, "Wording Changes from Ada 2012") -- "corrections" are identified that way, while amendments are not identified specially.

From: Simon Wright <simon@pushface.org>
Date: Thu, 06 Jul 2017 19:43:49 +0100
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> [...] Even if you use Latin-1, the set of allowed characters is defined as those that belong to NFKC.

I don't understand.

If your source has no BOM and you don't say -gnatW8, GNAT expects Latin-1 encoding. If your source has a BOM or you say -gnatW8, GNAT expects UTF8 encoding (I haven't tried what happens if you use NFD).

I haven't tried giving UTF8 coding without BOM or -gnatW8 - ignoring the use in unit names - ARM 2.1(16) says it should be accepted.

(later) UTF8 is accepted in strings but not in identifiers.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 7 Jul 2017 10:26:08 +0200
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> [...] I don't understand. [...]

This is a common confusion between characters, coded sets, and encodings...

ISO-10646 defines a coded set (code points) for a number of characters (identical to the one defined by Unicode). Some of these characters can be represented in NFKC. These are the allowed characters.

If you use Latin-1, you have different code points for the same characters - and the allowed characters are still those representable in NFKC, even with different code points.

UTF8 is an encoding, nothing more than a compression algorithm for numerical values. It is generally used to compress Unicode strings, but could be used for any numerical values. In any case, it doesn't change logical values, just the way they are stored.

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Fri, 07 Jul 2017 13:49:57 +0200
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> The rest is about GNAT's behaviour; to reiterate, ARM 2.1(16/3) says

> "An Ada implementation shall accept Ada source code in UTF-8 encoding, with or without a BOM (see A.4.11), where every character is represented by its code point."

> which for GNAT is not met unless either there is a BOM or -gnatW8 is used.

Which sounds perfectly okay.

There are no limitations to which command-line arguments a program can require to behave like an Ada compiler.

> On the other hand, ARM 2.1(4/3) says "The coded representation for characters is implementation defined", which seems to conflict with (16) - but then, the AARM ramification (4.b/2) notes that the rule doesn't have much force!

That sounds like the classical wording.

I suppose that the intent is that UTF-8 encoded ISO-10646 (in the right normalization form) _has_ to be supported, but that any other encoding is allowed in addition to that.

It would of course be nice if that was also what the ARM actually said.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 7 Jul 2017 14:44:17 -0500
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> I suppose that the intent is that UTF-8 encoded ISO-10646 (in the right normalization form) _has_ to be supported, but that any other encoding is allowed in addition to that.

Precisely.

> It would of course be nice if that was also what the ARM actually said.

Mostly we're not changing text that doesn't have to be changed. In some cases, it would make more sense if it was changed, but since every change has a potential for errors and unintended consequences, its often best to leave stuff alone. (There are many cases where a "simple" change broke something else, leading to repeated fixes.)

Smart Pointers and Tagged Type Hierarchies

From: Felix Krause <contact@flyx.org>
Date: Mon, 24 Jul 2017 17:41:37 +0200
Subject: Smart Pointers and Tagged Type Hierarchies
Newsgroups: comp.lang.ada

With Ada's controlled types, it is possible to implement smart pointers which manage heap objects with reference-counting. There is more than one tutorial showing how that works.

A problem I encounter is how this can be used with type hierarchies i.e. I have a smart pointer managing a tagged type, and I want to be able to derive from that tagged type and still be able to use my smart pointer with that new type. Let me give an example: Assume I want to implement an abstract type Stream that represents a stream of events (has nothing to do with Ada.Streams). I will use a slightly modified Rosen '95 name scheme here for clarity: Reference is the smart pointer, Instance is the actual object. Let this be the base type:

```
package Stream is
   type Reference is new
         Ada.Finalization.Controlled
         with private;
   type Instance is abstract tagged
         limited private;
   type Instance_Pointer is
         access all Instance'Class;
--  Reference-counting implementation here
   overriding procedure Adjust (
         Object : in out Reference);
   overriding procedure Finalize (
         Object : in out Reference);
   --  Fetches an event from the stream
   procedure Fetch (
         Object : in out Instance;
         Ret    :    out Event) is abstract;
--  Initialize the smart pointer with an object.
--  The smart pointer takes control of that
-- object and will deallocate it when reference
-- count reaches zero.
   procedure Init (
          Object : in out Reference'Class;
          Impl   : in     Instance_Pointer);
   function Implementation_Access (
          Object : Reference'Class)
          return Instance_Pointer;
--  Is called before deleting the instance.
-- override if you have cleanup to do.
   procedure Finalize (
          Object : in out Instance) is null;
private
   type Reference is new
          Ada.Finalization.Controlled
          with record
      Impl : Instance_Pointer;
   end record;
   type Instance is abstract tagged
          limited record
      Refcount : Natural := 1;
   end record;
end Stream;
```

An example non-abstract type derived from this would be stream that reads events from a file:

```
package File_Stream is
   type Reference is new
          Stream.Reference with null record;
   procedure Init (
          Object : in out Reference;
          Path : String);
--  Fetches the current position within the file
   procedure Current_Position (
          Object : in out Reference;
          Line, Column : out Positive);
private
   type Instance is
          new Instance with record
      File : Ada.Text_IO.File_Access;
--  Possibly other fields, e.g. information
-- needed for Current_Position
   end record;
   --  Closes the file
   overriding procedure Finalize (
      Object : in out Instance);
end File_Stream;
```

Some observations:

- Unless all implementations are child classes of Stream, it is necessary to make the Instance type public.

- A derived type, if it wants to provide additional operations (like Current_Position), must not only derive from Instance, but also from Reference, to be able to provide an type-safe interface to those operations.

- As types derived from Stream possibly need to derive Stream.Reference, a consumer of a Stream object needs to take a Stream.Reference'Class as input. This type cannot be used for a record field, so I need to allocate it in heap memory and store a pointer if I want to memorize a Stream.Reference value anywhere.

- The implementation of Current_Position is cumbersome as I need the Implementation_Access function and convert the result to File_Stream.Instance, which creates a needless downcast check.

I think this is not an ideal interface for the user and I am searching for a better alternative. One thing I thought of is having a generic pointer, so that only the Instance is tagged:

```
package Stream is
   type Instance is
         abstract limited tagged private;
   --  Fetches an event from the stream
   procedure Fetch (
         Object : in out Instance;
         Ret : out Event) is abstract;
private
   type Instance is
          abstract tagged limited record
      Refcount : Natural := 1;
   end record;
end Stream;
generic
   type Implementation is
          new Stream.Instance with private;
package Stream.Smart is
   type Reference is
          new Ada.Finalization.Controlled
          with private;
   --  Reference-counting implementation
   overriding procedure Adjust (
          Object : in out Reference);
   overriding procedure Finalize (
          Object : in out Reference);
private
   type Implementation_Access is
          access all Implementation'Class;
   type Reference is
          new Ada.Finalization.Controlled
          with record
      Data : access Implementation_Access;
   end record;
end Stream.Smart;
```

This looks good at first glance. But now, all consumers of a Stream must also be generic and take an instance of the Smart package as generic parameter (at least if I want them to take the smart pointer and not Instance_Pointer as parameter, which is kind of the point).

Now I am wondering what others think of these approaches. Are there alternatives? Which one would be better from a user perspective?

From: Chris Moore <zmower@ntlworld.com>
Date: Mon, 24 Jul 2017 22:24:58 +0100
Subject: Re: Smart Pointers and Tagged Type Hierarchies
Newsgroups: comp.lang.ada

> [...] A derived type, if it wants to provide additional operations (like Current_Position), must not only derive from Instance, but also from Reference, to be able to provide an type-safe interface to those operations.

Why? All ops on Instance-derived types (including constructing subprograms) should be in terms of that type. References are for access only (ho ho).

> As types derived from Stream possibly need to derive Stream.Reference, a consumer of a Stream object needs to take a Stream.Reference'Class as input. This type cannot be used for a record field, so I need to allocate it in heap memory and store a pointer if I want to memorize a Stream.Reference value anywhere.

No. This way lies madness. A parallel hierarchy of References gains you very little and takes a lot of maintenance.

> The implementation of Current_Position is cumbersome as I need the Implementation_Access function and convert the result to File_Stream.Instance, which creates a needless downcast check.

The downcast has to go *somewhere*.

<snip generic version>

I had to do this kind of thing a great deal in the Ada version of the mal lisp interpreter. See for example:

https://github.com/zmower/mal/blob/master/ada/smart_pointers.ads

https://github.com/zmower/mal/blob/master/ada/types.ads

But my advice is to avoid tagged types if you can. They only make sense if your problem is wildly dynamic or you want to be lazy when thinking about memory allocation. mal qualified on both counts.

From: Felix Krause <contact@flyx.org>
Date: Thu, 27 Jul 2017 21:38:34 +0200
Subject: Re: Smart Pointers and Tagged Type Hierarchies
Newsgroups: comp.lang.ada

> Why? All ops on Instance-derived types (including constructing subprograms) should be in terms of that type. References are for access only (ho ho).

Well, if the smart pointer for File_Stream is the same type as the smart pointer for Stream, I'd need to downcast the retrieved access type each time I want to call a subroutine only defined for File_Stream, such as Current_Position in my example. An explicit downcast is undesirable because it implies that it may fail, which it cannot when I create a File_Stream reference with a construction subroutine and subsequently call Current_Position on it.

[...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 31 Jul 2017 23:07:49 -0500
Subject: Re: Smart Pointers and Tagged Type Hierarchies
Newsgroups: comp.lang.ada

> Well, if the smart pointer for File_Stream is the same type as the smart pointer for Stream, I'd need to downcast the retrieved access type each time I want to call a subroutine only defined for File_Stream, [...]

Our experience with Claw eventually led to an in-house rule that all access types (and pseudo-access-types like smart pointers would be the same) had to be access-to-classwide. Once we did that, we found that we could avoid most explicit type conversions by using dispatching; we did need a few in cases of operations only defined for a subset of child types.

<Rant>There are no "casts" in Ada. Moreover, talking about "up" or "down" when talking about type conversions is confusing, mainly because many computer people apparently have never seen an actual tree. I can speak from experience when I say that I've seen thousands of trees and almost every one of them had the roots on the bottom. (The rest were windfalls and had the roots on the side...:-) Ergo, "down" in a tree is inherently toward the roots. Since a lot of people like to draw their trees upside down, they're confused about where the root of a tree is. The only solution to that is to be explicit: convert toward the root or toward the leaves. The only thing a "downcast" should be used for is fly fishing (and I *hate* fishing :-). If you want people to understand, "cast" out that terminology.</Rant>

From: Christoph Karl Walter Grein <christ-usch.grein@t-online.de>
Date: Fri, 28 Jul 2017 02:21:53 -0700 (PDT)
Subject: Re: Smart Pointers and Tagged Type Hierarchies
Newsgroups: comp.lang.ada

I haven't followed the thread in detail, but perhaps this page can help you:

http://www.christ-usch-grein.homepage.t-online.de/Ada/Smart_Pointers.html

----

From: Emmanuel Briot <briot@adacore.com>
Date: Sun, 30 Jul 2017 12:45:48 -0700 (PDT)
Subject: Re: Smart Pointers and Tagged Type Hierarchies
Newsgroups: comp.lang.ada

I like the approach with Implicit_Dereference for the accessor In AdaMagica's link. In fact, I am pretty sure we could use this for the smart pointer itself, so that the call to Get at the bottom of the page is not even necessary. I started playing with that for GNATCOLL.Refcount, but did not have time to finish yet.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 31 Jul 2017 22:43:37 -0500
Subject: Re: Smart Pointers and Tagged Type Hierarchies
Newsgroups: comp.lang.ada

> [...] I want to be able to derive from that tagged type and still be able to use my smart pointer with that new type.

What's needed is something like "co-derivation", where multiple types are derived in lock-step. Sadly, this is effectively impossible in an OOP environment, because dispatching calls would have the wrong parameter types for any co-derived types. (I've spent quite a bit of effort in trying to make such a solution work, and have enlisted others to help, but the conclusion was that it wasn't promising.)

One would have to have some form of multiple dispatch to get around that, but that's a bridge too far for a language that's mainly used in embedded, safety-critical systems.

One could make it work for non-tagged types, but of course that prevents dispatching and extension. So it's probably not worth the effort to design.

Proposal: Smart Pointers

From: Alejandro R. Mosteo <alejandro@mosteo.com>
Date: Thu, 31 Aug 2017 14:12:17 +0200
Subject: Interest in standard smart pointers for Ada 2020
Newsgroups: comp.lang.ada

I wonder if there would be interest in standardizing some usual smart pointers for the next revision. I will try to state the problems. I hope you'll point any misunderstandings on my part.

Ada has the particular limited and indefinite types, typically absent in other languages. However, their constraints often get in the way of comfort (all the cases I will list have workarounds, it's only tiresome and unnecessary noise):

- You cannot have indefinite record members, unless you make the record indefinite as well by providing a constraint. This basically moves the problem elsewhere (to the containing record, which may have no reason to be indefinite). Furthermore, some types do not have public constraints (e.g., to prevent declaration without initialization), so the previous solution wouldn't work. Also for class-wide members?

o Ada.Containers.Indefinite_Holders is aimed at this use case, but it is only for by-value semantics, so if you don't want to pay that penalty for some reason (large types) you're out of luck.

- You cannot have a limited type (obviously) as a member of an unlimited type. You're then forced to resort to low-level accesses or custom wrappers to pass around those members.
- Delayed initialization of limited types, specially in combination with indefinite-ness will require some access type use.
- Delayed initialization of indefinite types, when you don't want to/can have a default discriminant (I remember a recent discussion about limited-size indefinites).
- Any other use cases I'm forgetting about right now?

Arguably (I'm unsure about this) in most cases problems could be avoided with careful design? I don't know, but I do know that there are smart pointer Ada libraries around, and I have rolled my own more often than not, and when I try to go the unconstrained way all around, sooner or later I have to backpedal.

If we look at the cousin C++, the standard pointers there are:

> unique_ptr [1]

> Allows exactly one owner of the underlying pointer. Use as the default choice for POCO unless you know for certain that you require a shared_ptr. Can be moved to a new owner, but not copied or shared.

> shared_ptr [2]

> Reference-counted smart pointer. Use when you want to assign one raw pointer to multiple owners, for example, when you return a copy of a pointer from a container but want to keep the original. The raw pointer is not deleted until all shared_ptr owners have gone out of scope or have otherwise given up ownership.

> weak_ptr [3]

> Special-case smart pointer for use in conjunction with shared_ptr. A weak_ptr provides access to an object that is owned by one or more shared_ptr instances, but does not participate in reference counting. Use when you want to observe an object, but do not require it to remain alive. Required in some cases to break circular references between shared_ptr instances.

I'd argue for having these, and also thread-safe implementations (that I guess would be heavier, requiring protected internals). I'd be willing to work on this too, if there is interest.

[1] http://en.cppreference.com/w/cpp/memory/unique_ptr

[2] http://en.cppreference.com/w/cpp/memory/shared_ptr

[3] http://en.cppreference.com/w/cpp/memory/weak_ptr

[See also “Community Input for the ARG” earlier in this issue. —sparre] ptc® apexada | ptc® objectada®

Complete Ada Solutions for Complex Mission-Critical Systems

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

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

ptc

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.

2017

October 02-06 — 17th **International Conference on Formal Methods in Computer-Aided Design** (FMCAD'2017), Vienna, Austria. Topics include: theory and applications of formal methods in hardware and system verification; synthesis and compilation for computer system descriptions, modeling, specification, and implementation languages, model-based design, correct-by-construction methods, ...; experience with the application of formal and semi-formal methods to industrial-scale designs; tools that represent formal verification enablement, new features, or a substantial improvement in the automation of formal methods; etc.

☺ October 04-06 — 25th **International Conference on Real-Time Networks and Systems** (RTNS'2017), Grenoble, France. Topics include: real-time applications (automotive, avionics, process control, multimedia, cyber-physical systems, ...); software technologies for real-time systems (compilers, programming languages, middleware, RTOS, ...); real-time system design and analysis (real-time scheduling, mixed-criticality systems, model-driven development, WCET estimation, ...); formal specification and verification (formal methods, model checking, ...); real-time distributed systems (fault tolerance, task/messages allocation, IoT, ...); etc.

October 09-11 — 18th **International Conference on System and Design Languages** (SDL'2017), Budapest, Hungary. Topics include: evolution of development languages (domain-specific language profiles; modular language design; language extensions, semantics and evaluation; real-time aspects and performance; methodology for application; education and promotion); model-driven development; industrial application reports (industrial usage reports; standardization activities; tool support and frameworks; domain-specific applicability, such as automotive, aerospace, control, ...); etc.

☺ October 11-13 — 30th **International Workshop on Languages and Compilers for Parallel Computing** (LCPC'2017), College Station, Texas, USA. Topics include: compilers for parallel computing, parallel programming models and languages, formal analysis and verification of parallel programs, debugging tools for concurrency, concurrent data structures, parallel applications, software engineering for parallel programs, etc.

October 15-20 — **Embedded Systems Week** 2017 (ESWEEK'2017), Seoul, South Korea. Topics include: all aspects of embedded systems and software. Includes CASES'2017 (International Conference on Compilers, Architecture, and Synthesis for Embedded Systems), CODES+ISSS'2017 (International Conference on Hardware/Software Codesign and System Synthesis), EMSOFT'2017 (International Conference on Embedded Software).

> ☺ Oct 15-20 — ACM SIGBED **International Conference on Embedded Software** (EMSOFT'2017). Part of ESWEEK, EMSOFT brings together researchers and developers from academia, industry, and government to advance the science, engineering, and technology of embedded software development. EMSOFT is a venue for cutting-edge research in the design and analysis of software that interacts with physical processes, with a long-standing tradition for results on cyber-physical systems, which compose computation, networking, and physical dynamics.

October 15-20 **International Conference on Compilers, Architecture, and Synthesis for Embedded Systems** (CASES'2017). Part of ESWEEK, CASES is a forum where researchers, developers and practitioners exchange information on the latest advances in compiler and architectures for high-performance, low-power embedded systems. The conference has a long tradition of showcasing leading edge research in embedded processor, memory, interconnect, storage architectures and related compiler techniquest targeting performance, power, predictability, security, reliability issues for both traditional and emerging application domains. In addition, we invite innovative papers that address design, synthesis, and optimimization challenges in heterogeneous and accelerator-rich architectures.

October 19 7th **International Workshop on Design, Modeling and Evaluation of Cyber Physical Systems** (CyPhy'2017). In conjunction with ESWEEK 2017. Topics include: modeling and simulation languages for hybrid and cyber-physical systems; development of industrial or research-oriented cyber-physical systems in domains such as robotics, smart systems (homes, vehicles, buildings), medical and healthcare devices, future generation networks; evaluation of novel research tools, comparisons of state of the art tools in industrial practice; etc.

☺ October 22-27 ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2017), Vancouver, Canada. Topics include: all aspects of software construction, at the intersection of programming, languages, systems, and software engineering.

October 23-25 3rd **Symposium on Dependable Software Engineering: Theories, Tools and Applications** (SETTA'2017), Changsha, China. Topics include: formalisms for modeling, design and implementation; model checking, theorem proving, and decision procedures; scalable approaches to formal system analysis; integration of formal methods into software engineering practice; contract-based engineering of components, systems, and systems of systems; formal and engineering aspects of software evolution and maintenance; parallel and multi-core programming; embedded, real-time, hybrid, and cyber-physical systems; mixed-critical applications and systems; safety, security, reliability, robustness, and fault-tolerance; applications and industrial experience reports; tool integration; etc.

October 23-26 28th IEEE **International Symposium on Software Reliability Engineering** (ISSRE'2017), Toulouse, France. Topics include: innovative, high-impact techniques and tools for assessing, predicting, and improving the reliability, safety, and security of software products; validation and verification, testing; faults, errors, failures, defects, bugs; software quality and productivity; software security; dependability, survivability, fault tolerance and resilience of software systems; systems (hardware + software) reliability engineering; supporting tools and automation; industry best practices; software standards; etc.

October 23-27 14th **International Colloquium on Theoretical Aspects of Computing** (ICTAC'2017), Hanoi, Vietnam. Topics include: principles and semantics of programming languages; models of concurrency, security, and mobility; real-time, embedded, hybrid and cyber-physical systems; program static and dynamic analysis and verification; software specification, refinement, verification and testing; model checking; case studies, theories, tools and experiments of verified systems; etc.

Oct 30 – Nov 03 32nd IEEE/ACM **International Conference on Automated Software Engineering** (ASE'2017), Urbana-Champaign, Illinois, USA. Topics include: foundations, techniques, and tools for automating the analysis, design, implementation, testing, and maintenance of large software systems; such as component-based systems; maintenance and evolution; model-driven development; reverse engineering and re-engineering; specification languages; software architecture and design; software product line engineering; software security and trust; testing, verification, and validation; etc.

November 07-09 30th IEEE **Conference on Software Engineering Education and Training** (CSEET'2017), Savannah, USA. Topics include: curriculum development; empirical studies; personal or institutional experience; software assurance, quality, and reliability; methodological aspects of software engineering education; open source in education; cooperation between industry and academia; etc.

November 09-10 11th ACM/IEEE **International Symposium on Empirical Software Engineering and Measurement** (ESEM'2017), Toronto, Canada. Topics include: strengths and weaknesses of software engineering technologies and methods from a strong empirical viewpoint, including quantitative, qualitative, and mixed studies; case studies, action research, and field studies; replication of empirical studies and families of studies; mining software engineering repositories; empirically-based decision making; assessing the benefits/costs associated with using certain development technologies; industrial experience, software project experience, and knowledge management; software technology transfer to industry; empirical studies with negative results, i.e. studies that did not deliver the expected results; etc.

☺ November 14-16 **International Conference on Reliability, Safety and Security of Railway Systems** (RSSRail'2017), Pistoia, Italy. Topics include: safety in development processes and safety management combined approaches to safety and security system and software safety analysis formal modelling and verification techniques system reliability validation according to the standards tool and model integration, toolchains domain-specific languages and modelling frameworks model reuse for reliability, safety and security modelling for maintenance strategy engineering etc. Deadline for early registration: October 2, 2017.

Nov 29 – Dec 01 18th **International Conference on Product-Focused Software Process Improvement** (PROFES'2017), Innsbruck, Austria. Topics include: experiences, ideas, innovations, as well as concerns related to professional software process improvement motivated by product and service quality needs; industrial experience reports and empirical studies reporting, e.g., on the application of respective methods or technologies in real settings; etc.

☺ December 05-08 38th IEEE **Real-Time Systems Symposium** (RTSS'2017), Paris, France. Topics include: all aspects of real-time systems theory, design, analysis, implementation, evaluation, and experiences.

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

2018

January 16-19 10th **Software Quality Days Conference** (SWQD'2018), Vienna, Austria. Theme: "Software Quality 4.0: Advanced Methods and Tools for better Software and Systems". Topics include: improvement of software development methods and processes; testing and quality assurance of software and software-intensive systems; domain specific quality issues such as embedded, medical, automotive systems; novel trends in software quality; etc.

Jan 29 – Feb 02 44th **International Conference on Current Trends in Theory and Practice of Computer Science** (SOFSEM'2018), Krems an der Donau, Austria. Topics include: foundations of computer science, software engineering, and data and knowledge-based systems. Deadline for submissions: November 20, 2017 (posters).

February 03-04 **Free and Open Source Software Developers' European Meeting** (FOSDEM'2018), Brussels, Belgium.

February 07-09 12th **International Workshop on Variability Modelling of Software-Intensive Systems** (VaMoS'2018), Madrid, Spain. Topics include: variability across the software life cycle; runtime variability approaches; variability in software architecture; managing variability at post-deployment time; formal verification, testing, and debugging of variable software systems; refactoring and evolution of variable software systems; Reverse engineering approaches; formal reasoning and automated analysis on variability; software economic aspects of variability; etc. Deadline for submissions: October 13, 2017 (abstracts), October 23, 2017 (papers).

February 21-24 49th ACM **Technical Symposium on Computer Science Education** (SIGCSE'2018), Baltimore, Maryland, USA.

March 19-22 24th **International Working Conference on Requirements Engineering - Foundation for Software Quality** (REFSQ'2018), Utrecht, the Netherlands. Deadline for submissions: October 2, 2017 (papers), October 16, 2017 (workshops), January 15, 2018 (workshop papers).

☺ April 09-12 **The Art, Science, and Engineering of Programming Conference** (Programming'2018), Nice, France. Topics include: everything to do with programming, including the experience of programming; general-purpose programming; distributed systems programming; parallel and multi-core programming; security programming; interpreters, virtual machines and compilers; modularity and separation of concerns; model-based development; testing and debugging; program verification; programming education; programming environments; etc. Deadline for submissions: October 1, 2017 (workshops), December 1, 2017 (research papers, deadline 3).

April 09-13 33rd ACM **Symposium on Applied Computing** (SAC'2018), Pau, France.

☺ April 09-13 **Track on Object-Oriented Programming Languages and Systems** (OOPS'2018). Topics include: aspects and components; code generation, and optimization; distribution and concurrency; evaluation; formal verification; Internet of Things technology and programming; integration with other paradigms; interoperability, versioning and software evolution and adaptation; language design and implementation; modular and generic programming; runtime verification and monitoring; safe, secure and dependable software; static analysis; testing and debugging; type systems; etc.

April 09-13 **Track on Software Verification and Testing** (SVT'2018). Topics include: new results in formal verification and testing, technologies to improve the usability of formal methods in software engineering, applications of mechanical verification to large scale software, model checking, correct by construction development, model-based testing, software testing, static and dynamic analysis, analysis methods for dependable systems, software certification and proof carrying code, fault diagnosis and debugging, verification and validation of large scale software systems, real world applications and case studies applying software testing and verification, etc.

April 09-13 9th ACM/SPEC **International Conference on Performance Engineering** (ICPE'2018), Berlin, Germany. Theme: "Continuous Performance Assurance in Agile Delivery". Deadline for submissions: October 13, 2017 (workshops), October 16, 2017 (research and industrial/experience abstracts), October 18, 2017 (tutorials, research and industrial/experience papers), December 15, 2017 (artifact registration), December 22, 2017 (artifact submission), January 3, 2018 (posters/demos), January 10, 2018 (work-in-progress/vision papers).

April 10-13 11th **Cyber-Physical Systems Week** (CPS Week'2018), Porto, Portugal. Deadline for submissions: November 3, 2017 (workshops, tutorials, competitions).

☺ April 11-13 24th IEEE **Real-Time and Embedded Technology and Applications Symposium** (RTAS'2018). In conjunction with CPSWeek'2018. Topics include: timing issues ranging from traditional hard real-time systems to latency-sensitive systems with soft real-time requirements; original systems and applications, case studies, methodologies and applied algorithms that contribute to the state of practice in the design, implementation and verification of real-time systems; embedded, networked and cyber-physical systems that consider real-time aspects; etc. Deadline for submissions: October 6, 2017.

April 11-13 9th ACM/IEEE **International Conference on Cyber-Physical Systems** (ICCPS'2018). In conjunction with CPSWeek'2018. Topics include: development of technologies, tools, and architectures for building CPS systems; design, implementation, and investigation of CPS applications; secure and resilient CPS infrastructure; etc. Deadline for submissions: October 6, 2017 (full papers).

April 14-21 21st **European Joint Conferences on Theory and Practice of Software** (ETAPS'2018), Thessaloniki, Greece. Events include: 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), SV-COMP (7th Competition on Software Verification). Deadline for submissions: October 13, 2017 (abstracts), October 20, 2017 (papers), November 22, 2017 (POST abstracts), November 24, 2017 (POST papers).

April 17-19 10th **NASA Formal Methods Symposium** (NFM'2018), Newport News, Virginia, USA. Topics include: identify challenges and provide solutions for achieving assurance for critical systems; model checking, static analysis, use of formal methods in software and system testing, compositional techniques, parallel and/or distributed techniques, safety cases and system safety, fault tolerance, model-based development, etc. Deadline for submissions: November 10, 2017 (abstracts), November 20, 2017 (papers).

♦ April 18-20 19th International Real-Time Ada Workshop (IRTAW'2018), Benicàssim, Spain. Deadline for submissions: February 4, 2018 (position papers).

April 30 – May 04 2nd **International Conference on Software Architecture** (ICSA'2018), Seattle, USA. Topics include: model driven engineering for continuous architecting; component based software engineering and architecture design; re-factoring and evolving architecture design decisions and solutions; architecture frameworks and architecture description languages; preserving architecture quality throughout the system lifetime; software architecture for legacy systems and systems integration; architecting families of products; software architects roles and responsibilities; training, education, and certification of software architects; industrial experiments and case studies; bold arguments against current research directions and results; results that challenge established results or beliefs giving evidence that call for fundamentally new

	directions, open up new research avenues where software architecture research can contribute; etc. Deadline for submissions: November 12, 2017 (workshops), January 18, 2018 (technical abstracts), January 25, 2018 (full technical papers), March 8, 2018 (New and Emerging Ideas, engineering track, Early Career Researchers Forum abstracts, workshop papers), March 9, 2018 (tutorials), March 15, 2018 (New and Emerging Ideas, engineering track, Early Career Researchers Forum papers).
May 21-23	17th **International Conference on Software Reuse** (ICSR'2018), Madrid, Spain. Theme: "New Opportunities for Software Reuse". Topics include: component-based reuse techniques, generative reuse, systematic reuse approaches helping industries transitioning from ad-hoc approaches, reverse engineering of potentially reusable components, evolution and maintenance of reusable assets, development of reusable components for Product Line Engineering, software variability approaches for configuring and deriving reusable assets, dynamic aspects of reuse (i.e post-deployment time), etc. Deadline for submissions: December 4, 2017 (papers).
May 21-25	32nd IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2018), Vancouver, Canada.
May 27 – June 03	40th **International Conference on Software Engineering** (ICSE'2018), Gothenburg, Sweden. Deadline for submissions: October 10, 2017 (workshops), October 15, 1027 (IEEE TCSE Harlan Mills Award nominations), October 23, 2017 (SE in Practice, SE Education and Training, SE in Society, New Ideas and Emerging Results), November 1, 2017 (technical briefings), November 20, 2017 (doctoral symposium, demos), January 8, 2018 (ACM Student Research Competition), January 15, 2018 (student contest on Software Engineering), January 22, 2018 (student volunteers), February 5, 2018 (posters).
June 11-15	30th **International Conference on Advanced Information Systems Engineering** (CAiSE'2018), Tallin, Estonia. Theme: "Information Systems in the Big Data Era". Topics include: methods, models, techniques, architectures and platforms for supporting the engineering and evolution of information systems and organizations in the big data era. Deadline for submissions: October 15, 2017 (workshops), November 24, 2017 (abstracts), December 1, 2017 (papers), March 4, 2018 (forum).
♦ June 18-22	23rd International Conference on Reliable Software Technologies - Ada-Europe'2018. Lisbon, Portugal. Sponsored by Ada-Europe. Deadline for submissions: January 22, 2018 (regular papers, industrial presentations, tutorials, workshops).
July 14-16	22nd **International Symposium on Formal Methods** (FM'2018), Oxford, UK. Topics include: formal methods for the engineering of computer-based systems and software.
☺ July 16-22	32nd **European Conference on Object-Oriented Programming** (ECOOP'2018), Amsterdam, the Netherlands.

"VectorCAST is unique in that it provides us with the ability to increase the reliability and quality of our flight software."

-- Honeywell

VECTOR software

vectorcast.com

VectorCAST is a TÜV SÜD Certified Software Tool for Safety Related Development

Vector Software, Inc.
Golden Cross House | 8 Duncannon Street | London WC2N 4JF UK | +44 203 603 0120 | sales@vectorcast.com

19th International Real-Time Ada Workshop – IRTAW 2018

Hotel Voramar, Benicàssim, Spain
18-20 April 2018
http://www.ada-europe.org/irtaw2018

Call for Papers

The International Real-Time Ada Workshop series has provided a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions, and has attracted participation from key members of the research, user, and implementer communities worldwide. Recent International Real-Time Ada Workshop meetings contributed to the Ada 2005/Ada 2012 standards, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar Tasking Profile.
In keeping with this tradition, the goals of the 19th edition of IRTAW will be to:

- Review Ada 2012 Issues *vis-a-vis* real-time systems;
- Examine experiences in using Ada 2012 for real-time systems and applications;
- Implementation approaches for Ada 2012 real-time features;
- Consider developing other real-time Ada profiles in addition to the Ravenscar profile;
- Implications to Ada with multiprocessors in development of real-time systems;
- Paradigms for using Ada for real-time distributed systems, with special emphasis on robustness as well as hard, flexible and application-defined scheduling;
- Analysis of specific patterns and libraries for real-time systems development in Ada;
- Ada in context of the certification of safety-critical and/or security-critical real-time systems;
- Examine the Real-Time Specification for Java and other languages for real-time systems development, their current implementations and their interoperability with Ada in embedded real-time systems;
- Industrial experience with Ada and the Ravenscar Profile in real-time projects;
- Consider the language vulnerabilities of the Ravenscar and full language definitions;
- Consider testing for compliance with the Real-Time Annex.

Participation at the 19th IRTAW is by invitation following the submission of a position paper addressing one or more of the above topics or related real-time Ada issues. Alternatively, anyone wishing to receive an invitation, but for one reason or another is unable to produce a position paper, may send in a one-page position statement indicating their interests. Priority will be given to submitted papers.

Submission Requirements

Position papers should not exceed ten pages in typical IEEE conference layout, excluding code inserts. All accepted papers will appear, in their final form, in the Workshop Proceedings, which will be published as a special issue of Ada Letters (ACM Press). Selected papers will also appear in the Ada User Journal. Authors with a relevant paper submitted to the 23rd International Conference on Reliable Software Technologies – Ada-Europe 2018 (deadline 24 January, 2018) may offer an extended abstract of the same material to IRTAW. Please submit position papers, in PDF format, to the Program Chair by e-mail: *brad.moore@shaw.ca*

Important Dates

Paper Submission: **4 February, 2018**
Notification of Acceptance: 23 February, 2018
Confirmation of Attendance: 9 March, 2018
Final Paper Due: 30 March, 2018
Workshop: April 18-20, 2018

Program Chair
Brad Moore, *General Dynamics Mission Systems, Canada*

Workshop Chair
Jorge Real *Universitat Politècnica de València, Spain*

AdaCore The GNAT Pro Company · Ada europe · DISCA DEPARTAMENT D'INFORMÀTICA DE SISTEMES I COMPUTADORS · UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Ada-Europe 2018

23rd International Conference on Reliable Software Technologies
18-22 June 2018, Lisbon, Portugal

Conference Chair
Nuno Neves
LASIGE/U. Lisboa

Program Chair
António Casimiro
LASIGE/U. Lisboa

Special Session Chair
Marcus Völp
University of Luxembourg

Tutorial and Workshop Chair
David Pereira
CISTER/ISEP

Industrial Co-Chairs
Marco Panunzio
Thales Alenia Space
José Rufino
LASIGE/U. Lisboa

Publication Chair
Pedro Ferreira
LASIGE/U. Lisboa

Exhibition Co-Chairs
José Neves
GMV
Ahlan Marriott
White Elephant GmbH

Publicity Chair
Dirk Craeynest
Ada-Belgium & KU Leuven

Local Secretariat
Madalena Almeida
Viagens Abreu S.A.

General Information

The **23rd International Conference on Reliable Software Technologies – Ada-Europe 2018** will take place in Lisbon, Portugal. 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

22 January 2018	Submission of papers, industrial presentation, tutorial and workshop proposals
9 March 2018	Notification of acceptance to all authors
24 March 2018	Camera-ready version of papers required
8 May 2018	Industrial presentations, tutorial and workshop material required

Topics

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

This edition of Ada-Europe features a focused **Special Session on Security in Safety-Critical Systems**. Safety-critical systems, on which we daily bet our lives, have become increasingly more complex, networked and distributed. In combination with the growing professionalism of adversarial teams, this demands for not only safe systems, but systems that also remain safe while under attacks. This session seeks (but is not limited to) contributions aiming at bridging the safety and security gap in cyber-physical and other safety-critical systems. The topics of interest include: Software and System Aspects of Secure and Dependable CPS, Vulnerabilities and Protective Measures for Safety-Critical System Infrastructures, and Fault and Intrusion Tolerance and Long-Term Unattended Operation for Safety-Critical Systems. For further information, please contact the Special Session Chair directly.

The topics of interest for the **general track of the conference** include, but are not limited to (full list on the website): Real-Time and Embedded Systems, Mixed-Criticality Systems, Theory and Practice of High-Integrity Systems, Software Architectures, Methods and Techniques for Software Development and Maintenance, Formal Methods, Ada Language and Technologies, Software Quality, Mainstream and Emerging Applications, Experience Reports in Reliable System Development, Experiences with Ada.

http://www.ada-europe.org/conference2018

Call for Regular and Special Session Papers

Authors of papers that are to undergo peer review for acceptance are invited to submit original contributions by 22 January 2018. Paper submissions shall be 14 LNCS-style pages in length. Authors for both the general track and the special session shall submit their work via EasyChair at *https://easychair.org/conferences/?conf=adaeurope2018*. The format for submission is solely PDF.

The International Conference on Reliable Software Technologies is listed in DBLP, SCOPUS and Web of Science Conference Proceedings Citation index, Google Scholar, and Microsoft Academic Search, among others.

Proceedings

The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer, and will be available at the conference. The authors of accepted regular and special session papers shall prepare camera-ready submissions in full conformance with the LNCS style, strictly by 24 March 2018. 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.

Call for Industrial Presentations

The conference seeks industrial presentations that deliver value and insight but may not fit the selection process for regular papers. Authors are invited to submit a presentation outline of at least 1 page in length by 22 January 2018. Submissions shall be made via EasyChair following the link *https://easychair.org/conferences/?conf=adaeurope2018*. 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 8 May 2018, 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.

Awards

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

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 Tutorial and Workshop 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 Co-chairs 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.

Venue

The conference venue is the VIP Executive Art's Hotel (left image), in the Parque das Nações area (central images), in Lisbon, Portugal. June is full of events in Lisbon, including the festivities in honour of St. António, with music, grilled sardines and popular parties taking place in the old neighbourhoods of Alfama and Bairro Alto, downtown (image on the right). Plan in advance! It is absolutely worth it!

The Ada High-Integrity Rapporteur Group (HRG) Status Report

Joyce Tokar

Chair of the High-Integrity Rapporteur Group
Pyrrhus Software, LLC, PO Box 1352, Phoenix, AZ 85001-1352, USA; Tel: +1 480-951-1010;
E-mail: tokar@pyrrhusoft.com

1 Introduction

The Ada High-Integrity Rapporteur Group (HRG) was established in 1995 to provide the content of Annex H: High Integrity Systems of the Ada Programming Language Standard (Ada) [1]. The work of the group has expanded to address more of the concerns that are arising in high integrity systems.

As the charter of the HRG states, the group will synthesize the essential requirements of typical sector-specific standards for high integrity applications which have a bearing on Ada and its supporting tools. Guidance will be developed for users, implementers, evaluators and certifiers. The guidance produced will be in a form suitable for reference in procurement.

2 Current Activities

The HRG produces technical reference documents for use in conjunction with Ada including:

- TR 24718 Guide for the Use of the Ravenscar Profile in High Integrity Systems [2]

  This Technical Report (TR) provides a complete description of the motivations behind the Ravenscar Profile, to show how conformant Ada programs using the profile can be analysed, and gives examples of usage. The Ravenscar profile is a subset of the Ada tasking model, restricted to meet the real-time community requirements for determinism, schedulability analysis and memory-boundedness. The Ravenscar profile is suitable for mapping to a small and eficient run-time system that supports task synchronization and communication. The profile has been designed such that the restricted form of tasking that it defines can be used for software that needs to be verified to the very highest integrity levels.

  This document was recently updated to address the 2012 changes and modifications of Ada. The revised document was submitted to ISO for approval and publication.

- TR 15942 Guidance for the Use of Ada in High Integrity Systems [3]

  This TR provides guidance on the use of Ada when producing high integrity systems. In producing such applications adherence to domain specific guidelines or standards has to be demonstrated to independent bodies. These guidelines and standards vary according to the application area, industrial sector, or nature of the risk involved.

  This TR assumes that a system is being developed in Ada to meet the criteria established in one of these domain specific standards. The primary goal of the document is to translate general requirements to Ada specific ones. The document provides guidance only; there are no “shall” statements. The TR identifies verification and validation issues which should be resolved and documented in accordance with the appropriate domain specific standard.

  This document is being updated to reflect changes in Ada as well as new domain specific standards and guidance that have been introduced since the publication of the TR. The revised TR is expected to be submitted to ISO for approval and publication in 2018.

- TR-24772-2 Guidance to Avoiding Vulnerabilities in Programming Languages – Vulnerability Descriptions for the Programming Language Ada [4]

  This Technical Report specifies Ada vulnerabilities to be avoided in the development of systems where assured behaviour is required for security, safety, mission-critical and business-critical software. In general, this guidance is applicable to the software developed, reviewed, or maintained for any application.

  Vulnerabilities described in this technical report document the way that the vulnerability described in the language-independent document TR 24773: Guide to avoiding vulnerabilities in programming languages through language selection and use [5] are manifested in Ada.

  This Technical Report is under development by the HRG and is expected to be completed by the end of 2017. The HRG will support activities associated with developing a similar document for SPARK, TR 24772-6 [6].

3 Conclusions

The HRG meets in conjunction with other Ada events such as the International Conference on Reliable Software Technologies. Work continues between meetings to enable the generation and update of TRs for the high integrity community.

The HRG is very active with a core set of members who continue to work as volunteers to provide guidance on the use of Ada in high-integrity systems. Please contact the convenor of this working group, Dr. Joyce L Tokar (tokar@pyrrhusoft.com), if you are interested in joining.

References

[1] ISO/IEC 8652:2012, Information technology – Programming languages – Ada, Edition 3, International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) Joint Technical Committee (JTC) 1 / SubCommittee 22 (ISO/IEC JTC 1/SC 22), 12/2012.

[2] ISO/IEC TR 24718:2015, Information technology – Programming languages – Guide for the use of the Ada Ravenscar Profile in high integrity systems, Edition 2, ISO/IEC JTC 1/SC 22, 02/2005.

[3] ISO/IEC TR 15942:2000, Information technology – Programming languages – Guide for the use of the Ada programming language in high integrity systems, Edition 1, ISO/IEC JTC 1/SC 22, 03/2000.

[4] ISO/IEC TR 24772-2, Information technology – Programming languages – Guide to avoiding vulnerabilities in programming languages – Part 2: Ada, Under Development, ISO/IEC JTC 1/SC 22.

[5] ISO/IEC TR 24773:2013 Information technology – Programming languages – Guide to avoiding vulnerabilities in programming languages through language selection and use, Edition 2, ISO/IEC JTC 1/SC 22, 03/2013.

[6] ISO/IEC TR 24772-2, Information technology – Programming languages – Guide to avoiding vulnerabilities in programming languages – Part 6: SPARK, Under Development, ISO/IEC JTC 1/SC 22.

Using GtkAda in Practice

Ahlan Marriott, Urs Maurer
White Elephant GmbH, Beckengässchen 1, 8200 Schaffhausen, Switzerland; email: software@white-elephant.ch

Abstract

This article is an extract from the industrial presentation "Astronomical Ada" which was given at the 2017 Ada-Europe conference in Vienna.

The presentation was an experience report on the problems we encountered getting a program written entirely in Ada to work on three popular operating systems: Microsoft Windows (XP and later), Linux (Ubuntu Tahr) and OSX (Sierra).

The main problem we had concerned the implementation of the Graphical User Interface (GUI). This article describes our work using GtkAda.

Keywords: Gtk, GtkAda, GUI

1 Introduction

The industrial presentation was called "Astronomical Ada" because the program in question controls astronomical telescopes.

1.1 Telescopes

The simplest of telescopes have no motor. An object is viewed simply by pointing the telescope at it. However, due to the rotation of the earth, the viewed object, unless the telescope is continually adjusted, will gradually drift out of view.

To compensate for this, a fixed speed motor can be attached such that when aligned with the Earth's axis it effectively cancels out the Earth's rotation.

However many interesting objects appear to move relative to the Earth, for example satellites, comets and the planets. To track this type of object the telescope needs to have two motors and a system to control them.

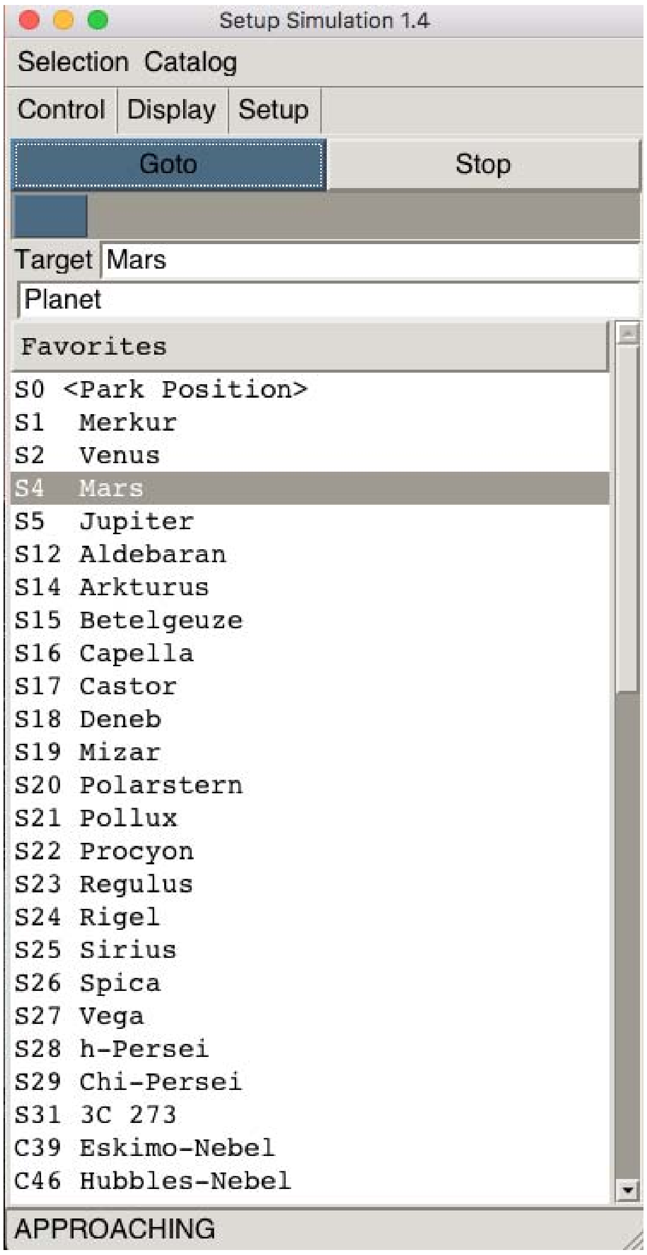
Using two motors the control system can position the telescope to view anywhere in the night sky.

Our Ada program (*SkyTrack*) is one such program. It can drive the motors to position the telescope onto any given object from within its extensive database and thereafter follow the object either by calculating its path or, in the case of satellites and comets, follow the object according to a downloaded pre-calculated path.

1.2 Graphical User Interface

The GUI is used to instruct the program where to position the telescope and what astronomical object it should follow.

The screen shot shown as figure 1 shows the *SkyTrack* program positioning the telescope on Mars. An object selected from the *Favourites* catalogue.



Target Mars
Planet

```
Favorites
S0 <Park Position>
S1  Merkur
S2  Venus
S4  Mars
S5  Jupiter
S12 Aldebaran
S14 Arkturus
S15 Betelgeuze
S16 Capella
S17 Castor
S18 Deneb
S19 Mizar
S20 Polarstern
S21 Pollux
S22 Procyon
S23 Regulus
S24 Rigel
S25 Sirius
S26 Spica
S27 Vega
S28 h-Persei
S29 Chi-Persei
S31 3C 273
C39 Eskimo-Nebel
C46 Hubbles-Nebel
```

APPROACHING

Figure 1 - SkyTrack GUI

The GUI was implemented using a package that provides a simple interface to create and manipulate common graphical objects. It was originally implemented using direct calls to the Windows API so, at least in theory; all we had to do was re-implement the implementation.

We chose to re-implement the GUI based on Gtk because both Gtk and Ada bindings to Gtk were available on all the designated target platforms.

GtkAda are Ada bindings to Gtk that are available from AdaCore at their web site libre.adacore.com/download. Unfortunately, by themselves, these are not sufficient to implement a GUI of any complexity. A lot of extra code has to be written in order that Gtk can actually be used.

This article describes the code we developed in order to use GtkAda.

2 Restrictions

The Windows API is not task safe. By which we mean that although the Windows *SendMessage* and *PostMessage* procedures are thread-safe, the API generally requires the passing of pointers to external objects. This is unsafe because the referenced object must be kept until the message is processed. Also the object must be locked against concurrent access because Windows supports message loops in different threads and the sending/posting of messages across thread borders.

Therefore in our original Windows based implementation we used protected objects to prevent concurrent API calls and an Ada task to process the Windows message loop.

However Gtk has the even more dramatic restriction that all Gtk calls **must** be executed from the **same** thread.

This required us to develop a system that provided our GUI with a simple and reliable means to make Gtk calls whilst at the same time guaranteeing that they were executed by the same thread.

In our implementation a dedicated thread is provided to process all the Gtk calls. The GUI package makes calls to this thread to request that it execute Gtk calls on its behalf.

In this arrangement, the GUI can be considered to be the client and the dedicated thread, the Gtk server.

We identified two types of Gtk request that the client may make: Synchronous and Asynchronous.

A synchronous request is a request made by the Gtk client to the Gtk server that expects the server to return a value. For example retrieving the contents of an edit box.

An asynchronous request is a request made by the Gtk client to the Gtk server that does **not** return a value. For example writing a row to a list view.

3 Synchronous requests

The synchronous interface consists of an abstract type and an abstract procedure based on this type.

```
type Request_Data is abstract tagged null record;
procedure Synchronous_Service (Data : in out
   Request_Data) is abstract;
```

The Gtk client makes a synchronous request to the Gtk server by extending the abstract type to include data that is to be sent to the server as well as the data that the client expects to receive from the server.

The following is an example demonstrating how to determine whether or not a specified check box is checked.

First the abstract type *Request_Data* is extended to make a new type *Check_Enquiry_Data*. This is defined to be a record containing two fields: *Check_Box* to specify the check box to be enquired and *Is_Checked* to hold the result of the enquiry.

```
type Check_Enquiry_Data is new Request_Data with
record
   Check_Box  : Gtk.Check_Button.Gtk_Check_Button;
   Is_Checked : Boolean;
end record;
```

The abstract procedure *Synchronous_Service* then has to be defined for the extended type. This procedure contains the code to be executed by the server on behalf of the client.

```
overriding procedure Synchronous_Service
  (Data : in out Check_Enquiry_Data) is
begin
   Data.Is_Checked := Data.Check_Box.Get_Active;
end Synchronous_Service;
```

The synchronisation and passing of data between the client and the server is implemented using a protected type that has two entries, one for the client to call and another that is used to block the client from immediately returning. The protected type also has a state and a means of retaining access to the client data.

```
type Request_Data_Ptr is access all
   Request_Data'class;
protected Gateway is
   entry Synchronous_Request (Data : in out
      Request_Data'class);
private
   entry Serviced (Unused_Data : in out
      Request_Data'class);
   State : Gateway_State := Idle;
   Data  : Request_Data_Ptr;
end Gateway;
```

In order that the client defined synchronous procedure is executed in the context of the server thread, the client needs to create a variable of the extended type, initialise it with the data required by the synchronous procedure and then rendezvous with the server.

At the rendezvous with the server the data will be passed to the server and the client blocked until the server has executed the synchronous procedure associated with the data.

In the following example the client function *Is_Checked* takes a check box as its only parameter. It puts this into a variable of type *Check_Enquiry_Data* that is an extension of *Request_Data* (see previously). The data is then passed to the Gtk server by making a rendezvous at the entry *Synchronous_Request*. When it is released from the entry it obtains the result from the variable and returns it to the caller.

```
function Is_Checked (The_Check_Box : Check_Box)
return Boolean is
   Data : Check_Enquiry_Data := (Request_Data with
      Check_Box  => The_Check_Box.The_Box,
      Is_Checked => False);
```

```
begin
   Gateway.Synchronous_Request (Data);
   return Data.Is_Checked;
end Is_Checked;
```

The entry *Synchronous_Request* within the protected type *Gateway* is implemented as follows:

```
entry Synchronous_Request (Data : in out
   Request_Data'class)
when State = Idle is
begin
   Gateway.Data := Data'unchecked_access;
   State := Busy;
   requeue Serviced;
end Synchronous_Request;
entry Serviced (Unused_Data : in out
   Request_Data'class)
when State = Ready is
begin
   State := Idle;
end Serviced;
```

Callers to *Synchronous_Request* are blocked until the server is ready to process the request by placing a guard on the entry, which is opened when the gateway state is set to *Idle*.

Within the entry a pointer is made to the data passed as the entry's parameter and the state set to *Busy*.

Finally it makes a call to the entry *Serviced* that effectively blocks the call from returning until the state is set to *Ready*.

In this way the client waits for the server to be *Idle,* sets up a pointer to the data, indicates that the data is ready and then waits for the server to indicate that it has processed the data.

Note that the requeue prevents the entry's parameter from being destroyed. Therefore until the state is set to *Ready*, the pointer *Gateway.Data* remains valid.

3.1 Synchronous Server

Making Gtk calls do not, by themselves, result in anything happening. For something to happen a thread must execute *Gtk.Main.Main_Iteration* in a loop.

```
loop
   Unused_Boolean := Gtk.Main.Main_Iteration;
end loop;
```

Consequently a minimum Gtk server must do this as well as process the synchronous requests made by the Gtk clients.

We can do this by modifying the Gtk *Main_Iteration* loop so that *Main_Iteration* is only called whilst there are Gtk events that need to be processed and then making a selective wait with timeout to check if there are any synchronous requests pending.

The code to determine whether there are any pending requests, to obtain the request and to signal that the request has been processed, is implemented by two entries and one function as part of the *Gateway* protected type.

```
protected Gateway is
   entry Check;
   entry Complete_Synchronous_Service;
end Gateway;
function Synchronous_Data return Request_Data_Ptr;
```

The entry *Check* blocks until the state is set to *Busy*. This happens after the client has entered *Synchronous_Request* and has made a pointer to the request data.

```
entry Check
when (State = Busy) is
begin
  null;
end Check;
```

The function *Synchronous_Data* can be used to access the request data.

```
function Synchronous_Data return Request_Data_Ptr
is
begin
   return Gateway.Data;
end Synchronous_Data;
```

The entry *Complete_Synchronous_Service* sets the state to *Ready* which frees the client blocked on the requeue at the entry *Serviced*.

```
procedure Complete_Synchronous_Service is
begin
   State := Ready;
end Complete_Synchronous_Service;
```

A Gtk server for synchronous requests can therefore be implemented as follows:

```
loop
   while Gtk.Main.Events_Pending loop
      Unused_Boolean := Gtk.Main.Main_Iteration;
   end loop;
   select
      Gateway.Check;
      Synchronous_Service
                   (Gateway.Synchronous_Data.all);
      Gateway.Complete_Synchronous_Service;
   or
      delay The_Period;
   end select;
end loop;
```

The server processes any pending Gtk events then checks for any client requests. If there aren't any within a short period of time (we typically wait for 50ms) the process is repeated.

If *Gateway.Check* is taken then the function *Synchronous_Data* is called to obtain the request data and then the client defined synchronous procedure associated with the data type is called. After which the procedure *Complete_Synchronous_Service* is called to release the client.

4 Asynchronous calls

We could have left it at that. We could have implemented our entire application by processing all our Gtk calls as synchronous requests. However if we had done so, the performance would have been very poor.

This is because the task switch between the client and the server and then back again are both relatively expensive. Using the synchronous mechanism to place a large amount of data into a list view is noticeably and unacceptably slow.

For the sake of efficiency we needed to implement an asynchronous method whereby the client can issue requests to the server to be processed at some future time. The client does not wait for the server to process these requests and therefore does not have to incur the penalty of the task switch back and forth to the server.

The asynchronous method is very similar to the synchronous method described previously, in so far that it relies on an abstract type that is extended to contain the request data and a procedure that overrides the type's abstract procedure.

It differs from the synchronous method in that instead of synchronising with the server it simply places the request into a protected queue ready for the server to process.

Unlike the synchronous method the client is not blocked and so is immediately free to make further requests. In the example of filling a list view with data, the client can first place all the requests into the queue and then, when this is done, the server can process the whole of the queue.

Having a queue of work to process avoids having to continually switch between client and server and is noticeably faster.

The asynchronous interface consists of an abstract type and an abstract procedure based on this type.

```
type Message_Data is abstract tagged null record;
procedure Asynchronous_Service
   (Message : Message_Data) is abstract;
```

The Gtk client makes an asynchronous request to the Gtk server by extending the abstract type to include data that is to be sent to the server.

The following is an example demonstrating how to set a specified check box.

First the abstract type *Message_Data* is extended to make a new type *Set_Check_Data*. This is defined to be a record containing the field *Check_Box* which is used to specify the check box to be set.

```
type Set_Check_Data is new Message_Data with
record
   Check_Box  : Gtk.Check_Button.Gtk_Check_Button;
end record;
```

The abstract procedure *Asynchronous_Service* then has to be defined for the extended type. This procedure contains the code to be executed by the server on behalf of the client.

```
overriding procedure Asynchronous_Service
  (Data : in out Set_Check_Data) is
begin
   Data.Check_Box.Set_Active (True);
end Asynchronous_Service;
```

The data is placed into the queue for the server to process using a protected type that has an entry and an indefinite doubly linked list that is used to implement the queue of requests.

```
package Message_List is new
   Ada.Containers.Indefinite_Doubly_Linked_Lists
   (Message_Data'class);
protected Gateway is
   procedure Asynchronous_Request (Data : in
      Message_Data'class);
private
   The_Messages : Message_List.Item;
end Gateway;
```

In order that the client defined asynchronous procedure is executed in the context of the server task, the client needs to create a variable of the extended type, initialise it with the data required by the asynchronous procedure and then place the data into the server queue.

In the following example, the client procedure *Set* takes as its only parameter the check box that should be set.

It copies the parameter into a variable of type *Set_Check_Data* that is an extension of *Message_Data* (see previously). The data is then placed into the Gtk server queue by making a call to the entry *Asynchronous_Request*.

```
procedure Set (The_Check_Box :
                     Gtk.Check_Button.Gtk_Check_Button)
is
  Data : Set_Check_Data
         := (Message_Data with
             Check_Box => The_Check_Box);
begin
   Gateway.Asynchronous_Request (Data);
end Is_Checked;
```

The entry *Asynchronous_Request* within the protected type *Gateway* is implemented as follows:

```
protected body Gateway is
   procedure Asynchronous_Request (Data : in
      Message_Data'class) is
   begin
      The_Messages.Append (Data);
   end Asynchronous_Request;
end Gateway;
```

4.1 Asynchronous Server

In order to process asynchronous requests, in addition to synchronous requests, the server needs to be extended. The *Check* entry needs to block until either a synchronous request is made or the queue of asynchronous work becomes not empty and for it to return what type of request is available.

```
type Data_Type is (Synchronous, Asynchronous);
entry Check (The_Data_Type : out Data_Type)
when not (State = Busy) or else
   (The_Messages.Count > 0) is
begin
   If The_Messages.Count > 0 then
```

```
      The_Data_Type := Asynchronous;
    elsif State = Busy then
      The_Data_Type := Synchronous;
    end if;
  end Check;
```

The main processing loop can then use the request type to decide how to process the request.

```
  loop
    while Gtk.Main.Events_Pending loop
      Unused_Boolean := Gtk.Main.Main_Iteration;
    end loop;
    select
      Gateway.Check (The_Data_Type);
     case The_Data_Type is
     when Synchronous =>
       Synchronous_Service
         (Gateway.Synchronous_Data.all);
       Gateway.Complete_Synchronous_Service;
     when Asynchronous =>
       Asynchronous_Service
         (Gateway.Next_Message);
       Gateway.Delete_First_Message;
     end case;
    or
      delay The_Period;
    end select;
  end loop;
```

The function *Next_Message* is a function to return the asynchronous request at the head of the asynchronous request queue and the procedure *Delete_First_Message* removes it from the queue.

```
  function Next_Message return Message_Data'class is
    The_Message : constant Message_Data'class :=
      The_Messages.First_Element;
  begin
    return The_Message;
  end Next_Message;

  procedure Delete_First_Message is
  begin
    The_Messages.Delete_First;
  end Delete_First_Message;
```

5 Callbacks

Most GUI implementations will require some form of callback in order that they can be notified of user interaction. Some callbacks need only identify the object (for example the button when a button is clicked) whilst others will require additional information (for example which row within a list view has been clicked).

GtkAda provides bindings to the Gtk mechanism however it is important to realise that only a very limited amount of work should be performed within these callbacks otherwise the responsiveness of the windowing system will be adversely affected.

To prevent this type of degradation, the Gtk callbacks in our GUI implementation are kept as simple as possible – any large amount of work is placed into a protected queue to be processed by a separate dedicated task.

For example, the Gtk callback called as a result of clicking on a button would add an action to the callback handler queue. The callback handler task processing this queue eventually processes the action; which invariably results in a routine being called that does whatever work is actually required.

By delegating this work to another task, the Gtk server task is released to service Gtk requests (perhaps generated as a result of the button being clicked) as well as processing the main Gtk event loop, thereby keeping windows and mouse tracking up to date.

Although the provision of this mechanism is not a requirement for a functional Gtk server, we found it convenient if the mechanism is provided in the same package as the server.

For example, causing *The_Action_Routine* to be executed whenever *The_Button* is clicked could be coded as follows.

```
  type Action_Routine is access procedure;
  package Action_Callback is new
    Gtk.Handlers.User_Callback (
      Widget_Type => Gtk.Widget.Gtk_Widget_Record,
      User_Type => Action_Routine);
  procedure Action_Handler (
    Unused : access
      Gtk.Widget.Gtk_Widget_Record'class;
    The_Action_Routine : Action_Routine) is
  begin
    Callback_Handling.Put (The_Action_Routine);
  end Action_Handler;
  Gtk.Button.Gtk_New (The_Button, “Button”);
  Action_Callback.Connect (
    The_Button,
    "clicked",
    Action_Callback.To_Marshaller(
      Action_Handler'access),
    The_Action_Routine);
```

When *The_Button* is clicked, Gtk calls the procedure *Action_Handler* in the context of the Gtk server thread. All this does is place *The_Action_Routine* into the callback queue.

A dedicated task *Callback_Handler* created by the Gtk server serially executes procedures placed in this queue. This can be coded as follows:

```
  package Callback_List is new
    Definite_Doubly_Linked_Lists (Action_Routine);

  protected Callback_Handling is
    procedure Put (The_Action : Action_Routine);
    procedure Finish;
    entry Get (The_Callback : out Action_Routine);
  private
    Is_Enabled : Boolean := True;
    The_Callback_List : Callback_List.Item;
  end Callback_Handling;
```

```
protected body Callback_Handling is
   procedure Put (The_Action : Action_Routine) is
   begin
      if Is_Enabled then
         The_Callback_List.Append (The_Callback);
      end if;
   end Put;
   procedure Finish is
   begin
      Is_Enabled := False;
   end Finish;
   entry Get (The_Routine : out Action_Routine) when
      (not Is_Enabled) or (The_Callback_List.Count > 0)
   is
   begin
      if Is_Enabled then
         The_Routine :=
            The_Callback_List.First_Element;
         The_Callback_List.Delete_First;
      else
         The_ Routine := null;
      end if;
   end Get;
end Callback_Handling;
task body Callback_Handler is
   The_Routine : Action_Routine;
begin
   loop
      Callback_Handling.Get (The_ Routine);
      exit when The_ Routine = null;
      The_Routine.all;
   end loop;
   The_Termination_Handler.Finalize;
end Callback_Handler;
```

5.1 Qualified callbacks

A qualified callback is a variation on the callback idea. It works in a similar fashion as the simple callback described previously but in addition returns client supplied data. This type of callback is used when it is insufficient just knowing which widget has been the subject of an event. For example when the row of a list view has been clicked the application invariably wants to know which row was clicked.

To support qualified callbacks we need to base the callback queue on a record that may contain different information depending on the type of the callback.

```
type Callback is (Simple, Qualified);
type Quaified_Routine is
   access procedure (Item : Information);
type Callback_Data (The_Callback : Callback := Action)
is record
   case The_Callback is
   when Simple =>
      Simple_Action : Action_Routine;
   when Qualified =>
      Qualified_Action : Qualified_Routine;
      The_Information : Information;
   end case;
end record;
package Callback_List is new
Definite_Doubly_Linked_Lists (Callback_Data);
```

and the protected type *Callback_Handling* extended accordingly.

```
protected Callback_Handling is
   procedure Put (The_Action : Action_Routine);
   procedure Put (The_Action : Qualified_Routine;
                  The_Information : Information);
   procedure Finish;
   entry Get (The_Callback : out Callback_Data);
end Callback_Handling;
task body Callback_Handler is
   The_Callback : Callback_Data;
begin
    loop
      Callback_Handling.Get (The_Callback);
      case The_Callback.The_Callback is
      when Simple =>
         exit when The_Callback.Simple_Action = null;
         The_Callback.Simple_Action.all;
      when Qualified =>
         The_Callback.Qualified_Action.all
            (The_Callback.The_Information);
      end case;
   end loop;
   The_Termination_Handler.Finalize;
end Callback_Handler;
```

The following is an example of how this extended callback mechanism could be used to indicate which row of a list view has been clicked.

```
package Qualified_Callback is new
Gtk.Handlers.User_Callback
(Gtk.Widget.Gtk_Widget_Record, Qualified_Routine);
Gtk.Tree_View.Gtk_New (The_View);
Qualified_Callback.Connect (
   The_View,
   "row-activated",
   Qualified_Callback.To_Marshaller
      (List_Click_Handler'access),
      The_Routine);
```

The procedure *List_Click_Handler* and the client supplied qualified routine are connected to the row activation event of the list view.

When the row of the list view is clicked, the procedure *List_Click_Handler* is called with both the list view widget and the user supplied qualified routine passed as parameters.

The *List_Click_Handler* procedure can then retrieve the information associated with the row that has been activated and then schedule a callback with this information.

```
procedure List_Click_Handler (
   Widget : access
      Gtk.Widget.Gtk_Widget_Record'class;
      The_Routine : Qualified_Routine)
is
```

```
   Iter  : Gtk.Tree_Model.Gtk_Tree_Iter;
   Model : Gtk.Tree_Model.Gtk_Tree_Model;
   Value : Glib.Values.GValue;
begin
   Gtk.Tree_Selection.Get_Selected
      (Gtk.Tree_View.Get_Selection
         (Gtk.Tree_View.Gtk_Tree_View(Widget)),
       Model, Iter);
   Gtk.Tree_Model.Get_Value (Model, Iter, 0, Value);
   Callback_Handling.Put (The_Routine,
      Information(Glib.Values.Get_Ulong(Value)));
end List_Click_Handler;
```

6 Closing the server

As previously described, the Gtk server sits in a loop either executing Gtk events or processing synchronous or asynchronous requests. However this is insufficient, we need a method to exit the loop and return control back to the thread that called the server procedure.

This is achieved by calling the *Gateway* entry *Quit*. The *Gateway* is the protected type used by clients to make synchronous or asynchronous requests. This needs to be enhanced so that a new type of request can be made.

This new type is the *Killed* request.

```
type Data_Type is (Killed, Synchronous, Asynchronous);
```

The gateway procedure *Quit* sets the gateway state to *Killed* and the *Check* entry is enhanced to return the *Killed* request if called in the *Killed* state.

```
procedure Quit is
begin
   State := Killed;
end Quit;
entry Check (The_Data_Type : out Data_Type)
when (State = Busy) or else
   (State = Killed) or else
   (The_Messages.Count > 0) is
begin
   if State = Killed then
      The_Data_Type := Killed;
   elsif The_Messages.Count > 0 then
      The_Data_Type := Asynchronous;
   elsif State = Busy then
      The_Data_Type := Synchronous;
   end if;
end Check;
```

The server can then use this request type as an indication that it should exit the otherwise infinite processing loop.

```
loop
   while Gtk.Main.Events_Pending loop
      Unused_Boolean := Gtk.Main.Main_Iteration;
   end loop;
   select
      Gateway.Check (The_Data_Type);
      case The_Data_Type is
      when Killed =>
         Gtk.Widget.Destroy
            (Gtk.Widget.Gtk_Widget(The_Main_Window));
         exit;
      when Synchronous =>
      …
      when Asynchronous =>
      …
      end case;
   end select;
end loop;
```

Note that this is also where the server destroys the main window that it created just before calling the start-up procedure.

7 The OSX restriction

Under MS-Windows and Linux, the server can be a standard Ada task; usually created in the package body. Unfortunately under OSX the thread that executes the Gtk calls **must** be the main thread of the process.

This tedious restriction means that the server has to be implemented as a procedure (which we call *Execute* – see below for details) that is called from the main thread and returns only when the GUI has been closed down.

To help synchronise start-up and shutdown we implemented our server procedure to include two procedures passed as parameters, one that is called immediately after the server is able to accept requests and the other that is called in response to the GUI being closed down.

Both of these procedures are executed in their own dedicated tasks so that they can better interact with the Gtk server.

Typically the start-up procedure is used to start GUI related tasks and to create the GUI objects, whilst the shutdown procedure is used to terminate these tasks.

```
procedure Execute (
   Startup_Routine  : access procedure
      (Window : Gtk.Window.Gtk_Window);
   Termination_Routine : access procedure);
```

7.1 Start-up

The server procedure creates the main Gtk window. It then creates the start-up task with the user supplied start-up procedure passed as its parameter. It then waits for the task to start by making a rendezvous with it, at which time it passes the previously created main Gtk window. After the rendezvous it executes the server code described previously.

```
task type Startup (Startup_Routine : access procedure
   (Window : Gtk.Window.Gtk_Window))
is
   entry Start (Window : Gtk.Window.Gtk_Window) ;
end Startup;
type Startup_Ptr is access Startup;
Startup_Task : Startup_Ptr;

task body Startup is
  The_Main_Window : Gtk.Window.Gtk_Window;
begin
   accept Start (Window : Gtk.Window.Gtk_Window) do
      The_Main_Window := Window;
   end Start;
```

```
      Startup_Routine.all (The_Main_Window);
   end Startup;
   procedure Execute (
      Startup_Routine  : access procedure
         (Window : Gtk.Window.Gtk_Window);
      Termination_Routine : access procedure)
   is
      The_Main_Window : Gtk.Window.Gtk_Window;
   begin
      Gtk.Window.Gtk_New (The_Main_Window);
      Startup_Task := new Startup (Startup_Routine);
      Startup_Task.Start (The_Main_Window);
      while Gtk.Main.Events_Pending loop
         Unused_Boolean := Gtk.Main.Main_Iteration;
      end loop;
      …  -- remainder of server
```

7.2 Termination

Early in the execution of the server procedure, a termination task is created.

```
   task type Termination_Handler is
      entry Start;
      entry Finalize;
   end Termination_Handler;
```

The server then connects a function to the delete-event of the main window.

```
   package Window_Callback is new
      Gtk.Handlers.Return_Callback
      (Gtk.Window.Gtk_Window_Record, Boolean);
      Window_Callback.Connect (The_Main_Window,
                                "delete-event",
                                Close_Window'access);
```

The connected function is called when the main window of the GUI is closed.

```
   function Close_Window ( Unused : access
      Gtk.Window.Gtk_Window_Record'class)
   return Boolean is
   begin
      The_Termination_Handler.Start;
      return True; -- Don't destroy the main window.
   end Close_Window;
```

This function starts the server termination task by making a rendezvous at its *Start* entry. Note that the function returns *True* to indicate to Gtk that the window should **not** be destroyed. This is so that the termination routine can still access the window in order that it can retrieve information before the window is actually closed and the information lost. For example its size and position on screen.

```
   task body Termination_Handler is
   begin
      accept Start;
      The_Termination_Routine.all;
      Callback_Handling.Finish;
      accept Finalize;
      Gateway.Quit;
   end Termination_Handler;
```

The termination task waits to be started then executes the termination routine supplied by the client. When this is finished it causes the callback task to terminate. It waits for the callback task to rendezvous at its *Finalize* entry to make sure that all the action routines have been executed before it finally closes the window and terminates the Gtk server by calling the Gateway *Quit* entry.

8 Downloads

A working example of a Gtk server package as described in this article may be downloaded from our web site www.white-elephant.ch

We cordially invite readers to comment and suggest improvements and/or corrections. We do not consider ourselves to be in any way knowledgeable with regard to Gtk or GtkAda and so would very much appreciate feedback.

Acknowledgements

We would like to acknowledge the work done by Dmitry A. Kazakov and Maxim Reznik in their GtkAda Contributions.
(http://www.dmitry-kazakov.de/ada/gtkada_contributions)

The work presented in this article was inspired by and derived from ideas implemented by Dmitry and Maxim in the afore-mentioned work.

However our implementation differs from theirs in that our Gtk server blocks (with timeout) waiting for requests rather than periodically processing pending requests using a timer and that our implementation of asynchronous requests uses a queue.

In our experience using a queue and a select with timeout greatly increases the performance of the GUI.

CubedOS: A Verified CubeSat Operating System

Carl Brandon, Peter Chapin, Chris Farnsworth, Sean Klink
Vermont Technical College, 201 Lawrence Place, Williston VT 05495, PO Box 500, Randolph Center, VT 05061; email: {cbrandon, pchapin}@vtc.edu

Abstract

In this paper we present CubedOS, a lightweight application framework for CubeSat flight software. CubedOS is written in SPARK*and proved free of certain classes of runtime errors. It consists of a collection of interacting, concurrent modules that communicate via message passing over a microkernel based on Ada's Ravenscar tasking model. It provides core services such as, for example, communication protocol processing and publish/subscribe message handling. Application-specific modules can be added to provide both high level functions such as navigation and power management, as well as low level device drivers for mission-specific hardware.*

Keywords: SPARK, student project, CubeSat

1 Introduction

CubedOS is being developed at Vermont Technical College's CubeSat Laboratory with the purpose of providing a robust software platform for CubeSat missions and of easing the development of CubeSat flight software. In many respects the goals of CubedOS are similar to those of the Core Flight Executive (cFE) written by NASA Goddard Space Flight Center [1]. However, unlike cFE, CubedOS is written in SPARK and verified to be free of the possibility of runtime error. SPARK has also been used to provide some other correctness properties in certain cases. We compare CubedOS and cFE in more detail in Section 4.

The intent is for CubedOS to be general enough and modular enough for many groups to profitably employ the system. Since every mission uses different hardware and has different software requirements, CubedOS is designed as a framework into which *modules* can be plugged to implement whatever mission functionality is required. CubedOS provides inter-module communication and other common services needed by many missions. CubedOS thus serves both as a kind of operating environment and as a library of useful tools and functions.

Some of the module functionality useful for complex CubeSat missions would include interfaces to attitude determination and control systems (ADACS), electrical power systems (EPS), photovoltaic panel orientation gimbals, navigation and data radio, data collection instruments, thermal control radiators, ion engine with gimbals, and cameras. We also plan on including a specific module for spiral thrusting which allows for three axis angular momentum control with a two axis thruster.

It is our intention that all CubedOS modules also be written in SPARK and at least proved free of runtime error. However, CubedOS allows modules, or parts of modules, to be written in full Ada or C. This allows CubedOS to take advantage of third party C libraries or to integrate with an existing C code base.

CubedOS runs on top of the Ada runtime system and thus works with any underlying platform supported by the available Ada compiler. For example, CubedOS makes use of Ada tasking without directly invoking the underlying system's support for threads. This simplifies the implementation of CubedOS while improving its portability. However, CubedOS does require that a rich Ada runtime system be available for all envisioned targets. Specifically, CubedOS requires a runtime system that supports the Ravenscar profile.

For resources that are not accessible through the Ada runtime system, CubedOS driver modules can be written that interact with the underlying operating system or hardware more directly. Although these modules would not be widely portable, they could, in some cases, be written to provide a kind of low level abstraction layer (LLAL) with a portable interface. We have not yet attempted to standardize the LLAL interface. However, we see that as an area for future work.

CubedOS applications are organized as a collection of active and passive modules, where each active module contains one or more Ada tasks. Passive modules do not contain any tasks but are used as containers for shared, reusable code. Although CubedOS is written in SPARK there need not be a one-to-one correspondence between CubedOS modules and SPARK packages. In fact, modules are routinely written as a collection of Ada packages in a package hierarchy.

Critical to the plug-and-play nature of CubedOS, each active module is self-contained and does not make direct use of any code in any other active module, although passive modules serving as library components can be used. All inter-module communication is done through the CubedOS infrastructure with no direct sharing of data or executable content. In this respect CubedOS active modules are similar to operating system processes. One consequence of this policy is that a library used by several modules must be either duplicated in each module, for example as private child packages, or provided as an independent, passive module. In this respect passive modules are similar to operating system shared libraries and have similar concerns regarding task safety and global data management.

In the language of operating systems, CubedOS can be said to have a microkernel architecture where task and memory management is provided by the Ada runtime system. Both low level facilities, such as device drivers, and high level facilities, such as communication protocol handlers, are all implemented as CubedOS modules. All modules are treated equally by CubedOS; any layered structuring of the modules is imposed by programmer convention.

CubedOS is currently a work in progress It is our intention to release CubedOS as open source once it is more mature and refined. We also need to review the code base to verify that it is free from International Traffic in Arms Regulations (ITAR) restrictions and possibly release both ITAR compliant and U.S non ITAR compliant versions. We anticipate this to happen in mid-2018.

2 CubedOS Architecture

To understand the context of the CubedOS architecture, it is useful to compare the architecture of a CubedOS application with that of a more traditional application. Since CubedOS is written in SPARK and must abide by the restrictions of Ravenscar, we compare CubedOS with other Ravenscar-based approaches.

Figure 1 shows an example application using Ravenscar tasking. Tasks, which must all be library level infinite loops, are shown as open circles and labeled as T_1 through T_4. Tasks communicate with each other via protected objects, shown as solid circles and labeled as PO_1 through PO_4.

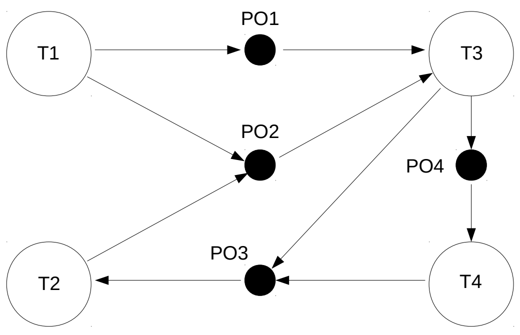


Figure 1: Traditional Ravenscar-based Architecture

Arrows from a sending task to a protected object indicate calls to a protected procedure to install information in the protected object. Arrows from a protected object to a receiving task indicate calls to an entry in the protected object used to pick up information previously stored in the object. Entry calls will block if no information is yet available but protected procedure calls do not block.

Ravenscar requires that protected objects have at most one entry and that at most one task can be queued on that entry. In CubedOS applications each protected object is serviced by exactly one task. This ensures that two tasks will never accidentally be queued on the protected object's entry. In the figure this means only one arrow can emanate from a protected object. However, multiple arrows can lead to a protected object, since it is permitted for many tasks to call the same protected procedure or for there to be multiple protected procedures in a given protected object.

In the example application of Figure 1, tasks T_1 and T_3 call protected procedures in two different protected objects. This presents no problems since protected procedures never block, allowing a task to call both procedures in a timely manner. However, task T_3 calls two entries, one in PO_1 and another in PO_2. Since entry calls can block, this means the task might get suspended on one of the calls leaving the other protected object without service for an extended time. The application needs to either be written so that will never happen or be such that it doesn't matter if it does.

There are several advantages over the traditional organization:

- The protected objects can be tuned to transmit only the information needed so the overhead can be kept minimal.
- The parameters of the protected procedures and entries specify the precise types of the data transfered so compile-time type safety is provided.
- The communication patterns of the application are known statically, facilitating analysis.

However the traditional architecture also includes some disadvantages:

- The protected objects must all be custom designed and individually implemented, creating a burden for the application developer.
- The communication patterns are relatively inflexible. Changing them requires overhauling the application.

A CubedOS application has an architecture as shown in Figure 2. In this case CubedOS provides the communication infrastructure as an array of general purpose, protected mailbox objects. CubedOS modules communicate by sending messages to the receiver module's mailbox. The messages are unstructured octet streams, and thus completely generic. Each active module has exactly one mailbox associated with it and contains a task dedicated to servicing that mailbox. That task extracts messages from the mailbox, and then decodes and acts on each message. Active CubedOS modules can also contain internal tasks as part of their implementation, but those tasks do not participate in message processing (although they can send messages) and are not important here.

The communication connections shown in Figure 2 are the same as those shown in Figure 1 except that the two communication paths from T_1 to T_3 are combined into a single path going through one mailbox.

CubedOS relieves the application developer of the problem of creating the communications infrastructure manually. Adding new message types is simplified with the help of a tool, XDR2OS3, that we describe in Section 3. In addition to providing basic, bounded mailboxes, CubedOS also provides other services such as message priorities and multiple sending modalities (for example, best effort versus guaranteed delivery). Many of these additional services would be tedious

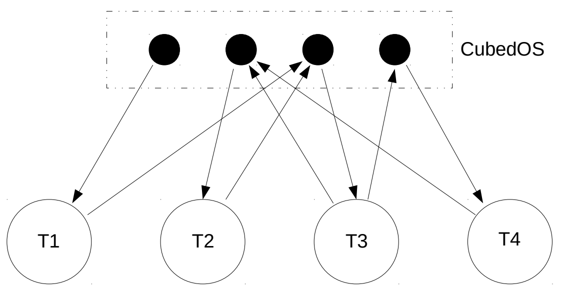


Figure 2: CubedOS-based Architecture

to provide on a case-by-case basis following the traditional architecture. CubedOS also allows any module to potentially send a message to any other module. Thus the communication paths in the running application are very flexible and dynamic.

Although the CubedOS architecture supports only point-to-point message passing, the CubedOS system provides an active module supporting a publish/subscribe discipline. The module allows multiple channels to be created to which other modules can subscribe. Publisher modules can then send messages to one or more channels, allowing for message broadcast and multicast. Since the messages themselves are unstructured octet streams, the publish/subscribe module can handle them generically without being modified to account for new message types.

Every CubedOS module has a statically assigned ID number. Messages sent from a module include the ID number of the sender. This allows a server module to return reply messages without statically knowing its clients. Thus server modules can be written as part of a general purpose "module library" and used without modification in a variety of applications. We have started compiling a registry of "well known" module IDs for common services, such as file handling and timer services. This allows CubedOS module libraries to make use of well known services and remain reusable. Here active CubedOS modules resemble network clients and servers where the module IDs play the role of a network address. Extending the architecture across different physical machines, or between different operating system processes on the same machine, is an interesting area for future work.

We are also defining standard message interfaces to certain services, such as file handling, that third party modules could implement. This allows modules to use a service without knowing which specific implementation backs that service.

However, CubedOS's architecture also carries some significant disadvantages as well:

- All mailboxes must have the same size since they are stored in an array. Some mailboxes will be larger than necessary, wasting space.
- All messages must have the same type and thus the same size. Some messages will be larger than necessary and slower to copy than necessary.

  The common message type also requires that typed information sent from one module to another be encoded into a raw octet format when sent, and decoded back into specifically typed data when received. This encoding and decoding increases the runtime overhead of message passing and reduces static type safety. Modules must defend themselves, at runtime, from malformed or inappropriate messages, causing certain errors that were compile-time errors in the traditional architecture to now be runtime errors. This is exactly counter to the general goals of high integrity system development.
- In order to return reply messages, the mailboxes must be addressable at runtime using module ID numbers. Accessing a statically named mailbox isn't general enough. As a result, the precise communication paths used by the system cannot easily be determined statically.

  In particular, since SPARK does not attempt to track information flow through individual array elements, it is necessary for us to manually justify certain SPARK flow messages. The architecture of CubedOS ensures that there is a one-to-one correspondence between a module and its mailbox. The tools don't know this, and the spurious flow messages they produce must be suppressed.

The details of CubedOS mitigate, to some degree, the problems above. For example, the mailbox array is actually instantiated from a generic unit by the application developer. This allows the developer to tune the sizes of the mailboxes, and the messages they contain, to the application's needs. CubedOS does not attempt to provide a one-size-fits-all mailbox array that will be satisfactory to all applications.

Also every well behaved CubedOS module should contain an |API| package with subprograms for encoding and decoding messages. This package is generated by the XDR2OS3 tool that we describe in Section 3. The parameters to these subprograms correspond to the parameters of the protected procedures and entries in the traditional architecture, and provide much of the same type safety. However, using the API subprograms is not enforced by the compiler. It is also possible to accidentally send a message to the wrong mailbox. Thus modules still need to include runtime error checking to detect and handle these problems.

So far we have described two extremes: a traditional approach that does not use CubedOS at all, and an approach that entirely relies on CubedOS. However, hybrid approaches are also possible. Figure 3 shows a combination of several CubedOS mailboxes and a hand-made, optimized protected object to mediate communication from T_3 to T_4.

This provides the best of both worlds. The simplicity and flexibility of CubedOS can be used where it makes sense to do so, and yet critical communications can still be optimized if the results of profiling indicate a need. In Figure 3 task T_4 can't be reached by CubedOS messages. The hand-made

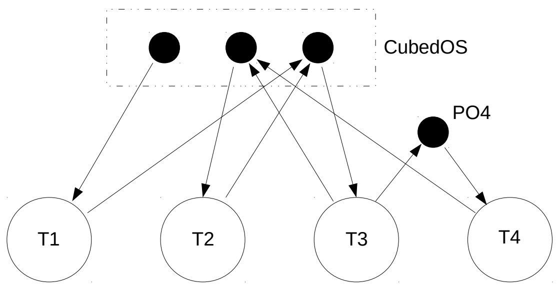


Figure 3: Hybrid Architecture

protected object creates a degree of isolation that can also simplify analysis as compared to a pure CubedOS system.

It is also possible to instantiate the CubedOS message manager multiple times in the same application, effectively creating multiple communication domains using separate mailbox arrays. Figure 4 shows an example of where T_4 is in a separate domain from the other modules (because it receives from a mailbox that is separate from the others).

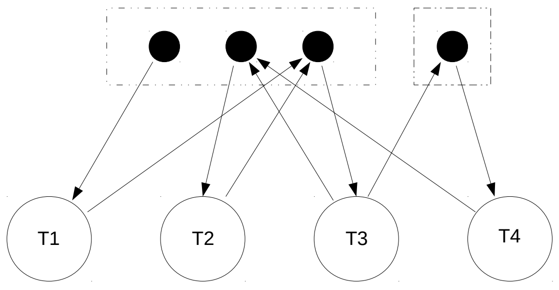


Figure 4: Multiple Communication Domains

This approach allows the CubedOS infrastructure to be used for easy development while still partitioning the system into semi-independent sections. For example, the sizes of the mailboxes and of the messages used in each communication domain need not be the same. The parts of the application that require large messages could be grouped into a domain separate from the parts that only require small messages.

Notice in Figure 4 tasks (modules) T_3 and T_4 send messages into multiple domains. This is, of course, sometimes necessary if the domains are going to interact. Modules that do this will need multiple module ID values scoped to different domains. At the moment the handling of this is largely a matter of manual configuration, which is reasonable for the relatively small programs typical of CubeSat missions. Creating a more comprehensive solution for module and domain addresses would be necessary as part of extending the architecture to multiple processes or machines as mentioned earlier. It is also likely that a naming service of some kind would need to be added to the module library provided by CubedOS. This is also an area for future work.

3 Message Encoding

CubedOS mailboxes store messages as unstructured octet arrays. This allows a general purpose mailbox package to store and manipulate messages of any type. Unfortunately this also requires that well structured, well typed message information be encoded to raw octets before being placed in a mailbox and then decoded after being retrieved from a mailbox.

The CubedOS convention is to use External Data Representation (XDR) encoded messages. XDR is a well known standard [2] that is also simple and has low overhead. We have defined an extension to XDR that allows SPARK's constrained scalar subtypes to be represented. We are currently working on a tool, XDR2OS3, that will compile a high level description of a message into message encoding and decoding subprograms. Our tool is written in Scala and is not verified, but its output is subject to the same SPARK analysis as the rest of the application. It is easier to prove the output of XDR2OS3 than it is to prove the correctness of XDR2OS3 itself.

The use of XDR2OS3 mitigates some of CubedOS's disadvantages. The developer need not manually write the tedious and repetitive encoding and decoding subprograms. Furthermore, those subprograms have well-typed parameters thus shielding the application programer from the inherent lack of type safety in the mailboxes themselves.

The use of XDR encoding may seem like an odd choice since XDR was originally defined for use in networking applications where data must be sent between potentially heterogenous systems. Since we envision current CubedOS applications being written entirely in SPARK and executing in a single process, XDR seems like a needless complication. However, as described in Section 2, we anticipate extending CubedOS to work in exactly the kind of potentially heterogenous environment XDR was developed to support. Thus we aim to provide a single standard for message encoding that will work in both the near and long term.

To illustrate CubedOS message handling, consider the following short example of a message definition file that is acceptable to XDR2OS3.

```
enum Series_Type { One_Shot, Periodic };

typedef unsigned int Module_ID
                     range 1 .. 16;
typedef unsigned int Series_ID_Type
                     range 1 .. 10000;

message struct {
    Module_ID      Sender;
    Time_Span      Tick_Interval;
    Series_Type    Request_Type;
    Series_ID_Type Series_ID;
} Relative_Request_Message;
```

This file introduces several types following the usual syntax of XDR interface definitions. The syntax is extended, however, to allow the programmer to include constrained ranges on the scalar type definitions in a style that is normal for Ada. The message itself is described as a structure containing various components in the usual way. The reserved word message prefixed to the structure definition, another XDR extension, alerts XDR2OS3 that it needs to generate encoding and decoding subprograms for that structure. Other structures serving only as message components (parameters) can also be defined.

XDR2OS3 has built-in knowledge of certain Ada private types such as Time_Span (from the Ada.Real_Time package). Private types need special handling since their internal structure can't be accessed directly from the encoding and decoding subprograms. There is currently no mechanism in XDR2OS3 to solve this problem in the general case.

Each message type has an ID number that is scoped by the module that defines the message. Upon receiving a message, the first step in message handling is to verify that the module ID of the receiver in the message header agrees with the ID of the module that is processing that message. This ensures that the message was actually sent to the intended module. Once that is done, the module is free to interpret the message ID value locally. Message ID values are never directly visible to module clients since the client calls a named encoding procedure to build each message. Thus the value and meaning of the message IDs defined by a module is entirely an implementation matter for the module. XDR2OS3 defines an enumeration type that specifies a module's messages as easy to read enumerators. It then uses the position value of a message enumerator as the message ID value.

XDR2OS3 is a work in progress. We intend to ultimately support as much of the XDR standard as we can including, for example, variable length arrays and discriminated unions. The development of XDR2OS3 is guided by our immediate needs with our currently envisioned missions, but we intend to extend and generalize the functionality of XDR2OS3 as the tool matures.

There are other possible encoding and decoding schemes that could have been used. For example, ASN.1 [3] is another standard with approximately similar goals as XDR. However, ASN.1 is much more complicated and entails more overhead both in space and time. ASN.1 includes type information in the encoded message itself, however, which may have advantages for error detection and handling. Since the application developer invokes tool-generated encoding and decoding procedures, and does not directly deal with message encoding, it would be possible to switch the message encoding method without significantly impacting applications. A future version of XDR2OS3 could potentially provide an ASN.1 mode (as one possiblity), perhaps for reasons of error handling or interoperability with legacy systems. This is also an area for future work.

4 Related Work

The most closely related work to CubedOS is NASA's Core Flight Executive [1]. Like CubedOS, cFE endeavors to be a general purpose framework for building flight software. Also like CubedOS, cFE is associated with a collection of modules, called the Core Flight System (CFS), that support common functionality needed by many missions. In addition, cFE makes use of a message passing discipline using a publish/-subscribe model of message handling. CubedOS can provide support for publish/subscribe message handling by way of a library module.

The main difference between the systems, aside from maturity level (cFE is a long established project with a history of actual use), is that CubedOS is written in SPARK and verified free of runtime error. In contrast, cFE is written in C with no particular static verification goals.

The cFE architecture is layered, whereas CubedOS modules operate as peers of each other. The cFE architecture makes use of a separate Operating System Abstraction Layer (OSAL) that presents a common interface across all the platforms supported by cFE. In contrast, CubedOS relies on the Ada runtime system for this purpose, and is thus Ada specific. CubedOS also uses a module library, the CubedOS LLAL, to provide hardware and OS independence in areas not covered by the Ada runtime system and standard library, but the interface to these modules is not yet standardized.

Kubos [4] is a project with roughly similar goals as cFE and CubedOS. It is not as mature as cFE. Like cFE, Kubos is written in C without static verification goals in the sense that we mean here.

Some CubeSat flight software is written directly on top of conventional embedded operating systems such as Linux, VxWorks [5], or RTEMS [6]. These systems allow application software to potentially be written with a variety of tools and methods, although C is most often used in practice. They provide flexibility by imposing few restrictions, but they also don't, by themselves, provide support for common flight software needs. Also they are themselves not statically verified as CubedOS is, although the Wind River VxWorks Cert platform [7] does provide a means by which VxWorks can be used in safety critical avionics applications conforming to the DO-178B standard.

5 Conclusion

CubedOS is an application framework based on message passing that is intended to support the flight software of space missions, particularly CubeSat missions. Our early experience with CubedOS is favorable. The architecture seems to provide an effective way to organize flight software.

Unlike similar projects such as cFE and Kubos, CubedOS is written entirely in SPARK and proved free of runtime errors in the sense meant by SPARK. It is necessary to manully suppress certain SPARK messages related to information flow through the CubedOS mailbox array. However, we feel the danger of doing this is minimal since the easy to understand architecture of the system ensures no flow problems will actually arise in practice.

CubedOS provides a great deal of concurrency and runtime flexibility, but sacrifices some static type safety to achieve this. We mitigate the danger using a tool, XDR2OS3, that generates message encoding and decoding subprograms based on strongly typed message descriptions. The output of the tool is verified by SPARK.

We intend to release CubedOS to the open source CubeSat community once we have completed an ITAR review of our code base and possibly release both ITAR compliant and U.S non ITAR compliant versions. We anticipate that release to be some time in mid-2018.

References

[1] "Core flight executive." `http://opensource.gsfc.nasa.gov/projects/cfe/`. Accessed 2017-01-22.

[2] M. Eisler, *RFC-4506: XDR: External Data Representation Standard.* Internet Engineering Task Force, May 2006. `http://tools.ietf.org/html/rfc4506.html`.

[3] ITU, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.* International Telecommunications Union, November 2008. `http://www.itu.int/rec/T-REC-X.680/en`.

[4] "Kubos." `http://www.kubos.co/`. Accessed 2017-01-22.

[5] "Vxworks." `http://www.windriver.com/products/vxworks/`. Accessed 2017-01-22.

[6] "Real time executive for multiprocessor systems." `https://www.rtems.org/`. Accessed 2017-01-22.

[7] "Wind river vxworks cert platform." `http://www.windriver.com/products/product-overviews/vxworks-cert-product-overview/`. Accessed 2017-01-22.

Parallelizing an Embedded Real-Time Person Matching System for Smart Cameras

Nesrine Abid, Kais Loukil, Walid Ayedi, Mohamed Abid

Laboratory of Computer and Embedded Systems, National School of Engineering of Sfax, 3038, Tunisia; Tel +216 23 217 951; email: nesrineabid88@gmail.com

Ahmed Chiheb Ammari

MMA Laboratory, National Institute of the Applied Sciences and Technology, 676 - 1080, Tunisia

Renewable Energy Group, Department of Electrical and computer engineering, Faculty of Engineering, King Abdulaziz University, 21589, Saudi Arabia

Abstract

Person matching is an important topic in video-surveillance and can be used to design detection, tracking and recognition systems. Multi-scale covariance (MSCOV) is considered as one of the most promising descriptors for person matching. Unfortunately, implementing such descriptors for person matching requires heavy computation. For a system that requires real-time matching, visual information needs to be fast processed without reducing accuracy. Parallel processing is frequently adopted to speed-up execution-time. This paper presents an optimized parallel model of a person matching system based on MSCOV. To this aim, a high-level parallelization approach based on the exploration of task and data levels of parallelism is adopted. Starting from the block diagram, an initial model that extracts the maximum task-level parallelism is proposed. This model is implemented and validated at a high system-level. Analyzing the communication and computation workload results, the potential bottlenecks of this model are identified. An optimized parallel model with the best workload balance is then developed. This model is prototyped and validated using a multicore architecture. The experimental results are promising and the system is shown to perform person matching in real-time with 16.33 fps using a dual-core ARM-Cortex-A9.

Keywords: multi-scale covariance descriptor, person matching, multi-core architecture, KPN parallel model.

1 Introduction

The development of smart camera networks is an emerging research field which represents the evolution of centralized computer vision applications towards distributed systems. In fact, in smart camera networks the logic of application is not centralized, but distributed among network nodes. Each node is capable to locally pre-process acquired images. A huge number of algorithms and techniques was developed targeting smart camera with various requirements to automatically detect, track and recognize person in video surveillance systems. Person matching has been an important topic in person detection, person tracking and person re-identification.

Person matching aims to determine a visual match between two image instances of the same individual appearing at different times and locations under different viewing conditions. Actually, person matching is a challenging problem because the imagery of a person may have strong variations due to the multiple geometrical transformations, deformations, appearance changes in the different viewing conditions. Different approaches have been used to tackle these problems. The performance of the person matching methods depends on the descriptor, the matching algorithm and the matching metric. Actually, a lot of interest has been given for person description based on characteristic feature descriptors like Haar-like features [1] Scale-Invariant Feature Transform (SIFT) [2], Histograms of Oriented Gradients (HOG) [3] and covariance descriptor (COV)[4]. Using such feature descriptors, person matching techniques are then implemented using Euclidean or Mahalanobis distance as a direct distance measure.

The SIFT descriptor extracts potential-points-of-interest and transforms them to an assigned scale, orientation and location for each feature. It is characterized by its invariance against the change of illumination, angle of view, scale and rotation. However, it has high computation complexity which limits its use. The Haar features are less complex. They define regions and compute in each one the difference between pixel intensities sums. However, their use is not recommended when image background is complex. The HOG descriptor has usually been considered as a standard method on video surveillance. It computes local intensity gradients and edge directions. The major inconvenience of HOG is the high dimension of the characteristic vector. Nevertheless COV descriptor [4] has been proved to be very effective for many computer vision tasks. The major advantage of COV is its ability to capture simultaneously the three aspects of features; shape, texture and color. These features are combined by measuring the variations and correlations between two features.

Experiments on [5] show that COV descriptor outperform HOG, SIFT and other known descriptors. The Multi-Scale Covariance descriptor (MSCOV) is an evolved presentation of COV descriptor that was originally proposed in [6].In this work, authors proposed to integrate a quadtree structure of the covariance matrix to capture only relevant information of the image and discard noise. MSCOV has been used in different contexts, including detection [7], matching [8], and re-identification [6]. In these works, authors proved that MSCOV largely outperforms COV and also many others well known descriptors as HOG and SIFT.A new matching technique adapted to the MSCOV descriptor is proposed in [8]. This technique can be used for designing robust systems for detection, tracking, and recognition.

Nevertheless, such video surveillance applications are computationally complex and time consuming. One of the most important design challenges of Smart Camera is to find the appropriate processing architecture capable to run complex and computationally intensive computer vision algorithms. For this aim, different processing solutions have been used. This includes the use of microprocessors CPUs [9], Digital Signal Processors DSPs [10], dedicated hardware implementations FPGA [11], Application-Specific Integrated Circuits ASICs [12] platforms. Traditionally, DSPs are ideally suited for intensive image signal processing. They generally integrate parallel processing capabilities, have internal memory blocks and provide multiplication and accumulation operations in a single cycle. However, DSPs have a high cost. The ASIC design can also offer high competitive performance, but they are specific, not flexible and not reconfigurable. ASIC devices are usually used only for high volume series manufacturing due to the high initial engineering cost of integrated circuits. FPGA devices can also meet real-time execution constraints thank to their high parallelization rate and data throughput. However, their exploitation requires significant code reformulations. Also translating computer vision library like OpenCV to hardware is a hard and time consuming task. Recently, with the increasing performance demands and the growing technological, new hardware architectures for embedded processing have emerged named multi-core systems. This technology consists of integrating many processing cores on the same chip. These systems can offer strong computational efficiency with high flexibility.

Theoretically, parallel architectures can help largely accelerate processing, but the gain obtained with application parallelizing is limited by data communications among processors, unbalance computation workload, memory access time, etc. In some cases, a parallel implementation may take longer to execute comparatively to its sequential version. This raises the need to develop efficient parallelization methodology that starts from the sequential application and provides a parallel model having the best processing and communication workload balance.

In this context, this paper proposes a high level parallel model for a matching technique application based on MSCOV. The Zynq-7000 platform based on dual core ARM Cortex A9 is adopted to evaluate the performance of the proposed parallel model. For this aim, a high-level parallelization approach is used. This approach is based on Khan Process Network (KPN) [13] models of computation implemented by Y-chart Applications Programmers Interface (YAPI) C++ library [14]. The key characteristic of this approach is the simultaneous use of data and task levels of parallelism to determine the parallel model that has the best processing and communication workload balance. Starting from the block diagram of the person matching system based on MSCOV descriptor, an initial model that extracts the maximum task-level parallelism is proposed. This model is validated at a high system-level using KPN and YAPI programming C++ runtime library. Analyzing the communication and computation workload results, the potential bottlenecks of the initial model are identified. Task level splitting, merging and data-level partitioning are then simultaneously explored to get a parallel model with the best computation and communication workload balance. Porting the proposed model on the Zynq-7000 platform, it is shown that comparatively to the original reference code, two times speedup of the system execution are obtained using a dual core ARM cortex A9 without losing any accuracy in the visual appearance performance.

The paper is organized as follows: Section 2 presents the person matching algorithm based on MSCOV descriptor. In section 3, the used parallelization approach is detailed and the different steps implemented to get an optimized parallel model are then discussed in section 4. The performance results of the obtained model are shown in section 5. Section 6 concludes the paper.

2 Person matching algorithm based on the Multi-Scale Covariance descriptor

Person matching is an important service on video surveillance systems. It saves a lot of human efforts on exhaustively searching and recognizing a person from large amounts of images and videos. It addresses the problem of person detection, person tracking and person re-identification. It consists in calculating the similarity between a reference object and its match at each position of the image. Initially a person region is defined as a reference object. Then matching of the reference object at all locations of the candidate image region is performed. The highest corresponding (matching) degree obtained at a given location is considered to be the best corresponded point. Typically, a person matching application includes three steps: extracting a discriminative feature as the person descriptor, calculating similarities between feature vectors and ranking the similarities that yields to a matched result. A sliding Window is used to create multiple candidate regions of the person. The window is a rectangular region that scans the entire image with a vertical and horizontal constant stride. Each window is described with a feature vector. The vector is calculated for the given region. Then, every vector is evaluated to predict if the region is the corresponding person or not.

The block diagram of the MSCOV person matching application is shown in Figure 1. It is constituted by four modules. The First module extracts features from the scene image. The second module creates the quadtree (IQF) and stores extracted features in the quadtree nodes. The third and fourth modules are two repetitive functions. The third module collect image quadrants of IQF which are inside each scanning widow and computing the covariance matrix. The fourth module calculates the Euclidian distance between the reference image and the scanned image window. The correctly matched region corresponds to the least measured Euclidian distance. More details about the application modules are given next.

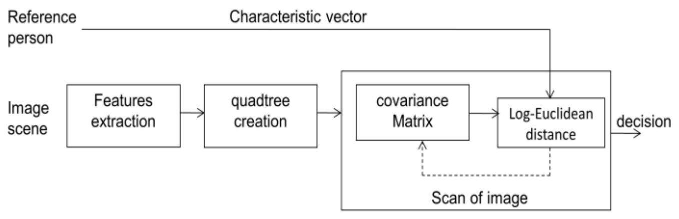


Figure 1 The block diagram of person matching based on MSCOV

2.1 Features extraction

MSCOV descriptors have been used in computer vision for person matching using ten features [6] which are a combination of structure features and content features. The structure features are the x location "X", the y location "Y" and the node level "l". The content features are the grayscale intensity value "I", the red chrominance component "Cr", the blue chrominance component "Cb", the norm of the first order derivatives in x "Ix", the norm of the first order derivatives in y "Iy", the Gradient "G" and the Magnitude "M" features. Structure features are related to the structure of the image representation. These features are therefore extracted when creating the quadtree. The color features can be computed in parallel. Once the "I" is extracted, the "Ix" and "Iy" can then also be computed in parallel. Using "Ix" and "Iy", the "G" and "M" features may also be computed simultaneously.

2.2 Quadtree creation

A quadtree is an image data structure which is represented by a set of nodes as presented in Figure. 2. Each node has four children. The whole image is represented by the root node. The root is recursively decomposed into four equal quadrants called nodes. This decomposition continues until a stopping condition of quadrant homogeneity is met. Each quadrant is associated to a node that is storing data about the corresponding quadrant. In the MSCOV descriptor, an image is presented by its quadree structure. Each node ‘c’ of the tree stores the extracted feature vector (Fc), the sum (Pc(k)) of each feature k at that node c, the multiplication (Qc(k, l)) of each two features k and l of the node c. All these features are used to compute the multi-scale covariance matrices (Cc). (Pc(k)) and (Qc(k, l)) of nodes whose children are leaves are computed using eq.1 and eq.2. (Pc(k)) and (Qc(k, l)) of parent nodes are computed from child ones using eq.3 and eq.4 With ci is the child nodes of parent node c.

$$P_C(k) = \sum_{i=0}^{3} F_{c_i}(k) \quad (1)$$

$$Q_C(k,l) = \sum_{i=0}^{3} F_{c_i}(k) \times F_{c_i}(l) \quad (2)$$

$$P_C(k) = \sum_{i=0}^{3} P_{c_i}(k)$$

(3)

$$Q_C(k) = \sum_{i=0}^{3} q_{c_i}(k,l) \quad (4)$$

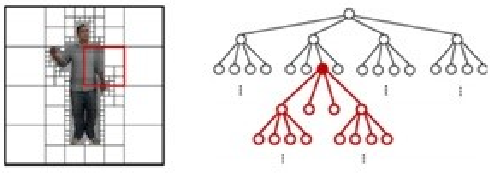


Figure 2 Quatree representation on MSCOV descriptor [6]

The multi-scale covariance matrices are computed by:

$$C_c = \frac{1}{N_c}\left[Q_c(k,l) - \frac{1}{N_c}P_c(k)P_c(l)\right] \quad (5)$$

where: N_c is the number of descendant nodes of C.

2.3 Similarity measures

The matching technique is based on the determining of the most similar candidate region of a reference object. The similarity search consists to find the minimum distance Δmin between a reference person image and all candidate regions of the processed image.

$$\Delta_{min} = \arg min \Delta(\varepsilon) \quad (6)$$

Where ε is the set of all candidate regions of an image.

In the MSCOV descriptor, a region is presented by quadrant node that is storing among other features a multi-scale covariance matrix. Therefore, to compare between two quadrant nodes, a Euclidian similarity measure of the two corresponding covariance matrices is required. Authors in [8] use the log Euclidian distance calculated by:

$$\Delta = \|\log(A) - \log(B)\| \quad (7)$$

With log(A) and log(B)are two log-arithmetic covariance matrices computed using the orthonormal matrix U and the eigenvalues λ_i by the following equation:

$$\log(A) = U.diag(\log(\lambda_1), \log(\lambda_2), \ldots. \log(\lambda_n)).U^T \quad (8)$$

3 The parallelization approach

Matching application based on MSCOV descriptor is computationally intensive and embedded implementation based on classical mono-processor architectures are often inadequate to achieve real time decoding performance superior to 10 frames per second (fps). To accelerate the system processing, some kind of multiprocessor implementation is motivated. Prior to any kind of multiprocessing, a parallel model of the person matching algorithm is needed. This paragraph starts by a literature review of previous developments of embedded systems targeting video surveillance applications, Kahn Processes Network model of computation and their implementation through the YAPI programming interface are then discussed. Finally, the approach used for the system parallelization is presented.

3.1 Motivation

Few embedded video surveillance systems are presented in the literature. Many of these devices are based on HOG descriptors using FPGA architectures [15]. To take advantage from the computing speed of hardware and the flexibility of software, authors in [16] proposed a HW/SW framework for person detection and tracking. They use a "Region Proposal Network" to extract detection regions of interest and HOG descriptor to track detected person. They used Zynq XC7Z020 SoC platform to prototyping their architecture composed by three hardware accelerators (Convolutional Neural Network, HOG and FFT) controlled by ARM. They reached 14.7 fps on person detection and 20 fps on tracking but they use image with low resolution (72 x72). Authors of [17] used Xilinx Spartan-6 LX-150T to design a hardware accelerator of HOG descriptor and ARM processor to compute the other functional modules of the detection application. Despite, 15 fps decoding rate are reached, a loss of the detection accuracy is observed. In [18], a software/hardware architecture for person detection based on HOC descriptor was designed using FPGA and ATOM processor. For this case, however, the obtained real-time detection performance is still below expectations, 2 times speedup was achieved compared to a full software mono processor implementation. In [19], a speedup of 3.22 times was achieved using ARM Coretex-A9 processor with a hardware accelerator but, the detection accuracy is decreased about 2.68%. In [20], a HOG-based pedestrian detection system is designed where a specific hardware accelerator is dedicated to the HOG module. For this case, real time detection performance is achieved however about 0.1–0.4% accuracy loss are observed in reference with full software implementation. Chen et al. [21] propose a low-cost high-speed hardware implementation of HOG where the detection accuracy is also shown to slightly decrease in reference to the performance of the original system. All of these HOG descriptor implementations are based on dedicated hardware accelerators, and are tolerating some performance accuracy decrease given the different algorithm modifications and simplifications used to ensure higher decoding rates.

In addition to embedded implementation based on HOG descriptors, there have been developments focusing on devoted accelerators of the covariance descriptor using FPGA. Martelli et al. [22] proposed a person detection frame work using covariance descriptor and machine learning classifier. The computation time is estimated to 92 fps VGA image. Authors in [23] proposed a HW/SW architecture of covariance descriptor. They used hardware accelerators to compute first and second integral features. A good performance was achieved by the hardwired covariance feature extractor. Authors on [15] proposed a framework for pedestrian detection based on sliding covariance matrix. They use Xilinx Virtex-6 FPGA platform to design a dedicated hardware covariance descriptor. Their proposed Hardware architecture is based on three modules: the features extraction module, the tensors calculation model and the covariance matrices calculation model. These models are applied on small image cells (block 16 x 16) and estimates execution time to 292 fps of VGA resolution.

Almost all of these previous works are using some kind of dedicated hardware accelerators to achieve real-time performance for person detection and tracking. These implementations do not offer enough flexibility to cope with adapting the system to different situations or to upgrade it with improved features and techniques. Also these works ignored the computing time and the connection of the rest modules required for detection.

A solution that provides higher flexibility while maintaining performance at appropriate levels is therefore desirable. Multi-core architectures are good alternatives. For multi-core processing systems, the reference application needs to be partitioned into multiple tasks that can be executed simultaneously on different cores. Generally for parallel computing, the most complex task is split into several parts that can be processed simultaneously. These parts are then distributed to diverse processors. A coordination mechanism should be employed to control the status of processors and to communicate with each other to produce final results. Prior to any multiprocessor implementation, a parallel specification is required to functionally describe the studied application as a set of processes exchanging data according to an appropriate model of computation. The predominant forms of parallelism are Data-Level Parallelism (DLP) and Task-Level Parallelism (TLP). DLP has been the most commonly used form of parallelism, implemented through vector or SIMD (Single Instruction Multiple Data) architectures [24]. This form of parallelism consists in extracting from the original source code regular data (vectors, matrices, etc.) on which the same sequence of instructions are applied. DLP is well adapted particularly to algorithms handling very large structures of homogenous data. On the other hand, TLP partitions the code by functionality. Each functionality is dedicated to execute on a separate execution unit. Optimized task-level decomposition needs to regroup some system functionalities to get a balanced task computation workload. An effective TLP implementation requires a data dependency analysis between tasks and an accurate study of their execution orders from the original sequential code. Since both TLP and DLP have their strengths and weaknesses, it is attractive to integrate both forms of parallelism to get an optimized model with the best computation workload and communication workload balance.

3.2 Kahn Processes Network model of computation

Kahn Process Networks (KPN) [13] has been used for modeling several signal processing applications. This model of computation assumes a network of concurrent autonomous processes that communicate over a point-to-point unbounded first-in-first-out (FIFO) channels, using a blocking-read and write synchronization primitives. Read actions from these FIFOs block until at least one data item becomes available. Write actions are non-blocking assuming the channels to have unbounded capacity. It is demonstrated that KPN is deterministic and independent of process interleaving, meaning that for a given input always the same output is produced and the same workload is generated, irrespective of the execution schedule [13]. Using KPN, a parallel model of an application can be specified in terms of distributed control and distributed memory which allows for its mapping onto a multiprocessing platform in a systematic and efficient way.

3.3 YAPI programming interface

YAPI [14] is a C++ library that implements KPN models of computation. A KPN/YAPI process network consists in a set of processes and a set of FIFO channels. Processes are provided with sets of input and output ports. A channel connects a process output port to a process input port of the same data type. Each port is connected to precisely one channel. A process gets blocked when it tries to read from an empty FIFO or it tries to write into a FIFO which is full. There are primitives provided by YAPI to ensure data transfer over channels. The “read” primitive is used to read data from an input port and to save it in a local variable within the process. The “write” primitive is used to write the value of a local variable into a FIFO channel. A validated YAPI network specification is platform independent and represents a valuable starting point for any multi-core implementation.

3.4 Approach used for the system parallelization

To get a matching parallel model based on MSCOV descriptor that has the best processing and communication workload balance, an appropriate approach has been used. The different steps of this approach are depicted in Figure 3.

The First step is "the system analysis". The entry for this step is a sequential C reference specification of the system. First, we start by optimizing as much as we can the original C reference code in terms of processing and memory usage. Computational and memory profiling are used to better understand the characteristics of the different application modules and identify the major system bottlenecks. The second step is "the task level parallelization" where the available task-parallelism is extracted by splitting compute nodes as far as possible. For this case, the application block diagram will serve as a starting point to extract the maximum task level parallelism. The original reference code is modified and structured by hand to describe the KPN in C++. Each Kahn process is described by a set of associated functions extracted from the original C code. The inter process communication is performed using YAPI FIFO primitives.

Using global variables is not allowed with a KPN model [13]. Thus, to ensure inter process communication, all the global shared variables used in the sequential reference code are grouped into associated data structures for communication over the FIFO channels

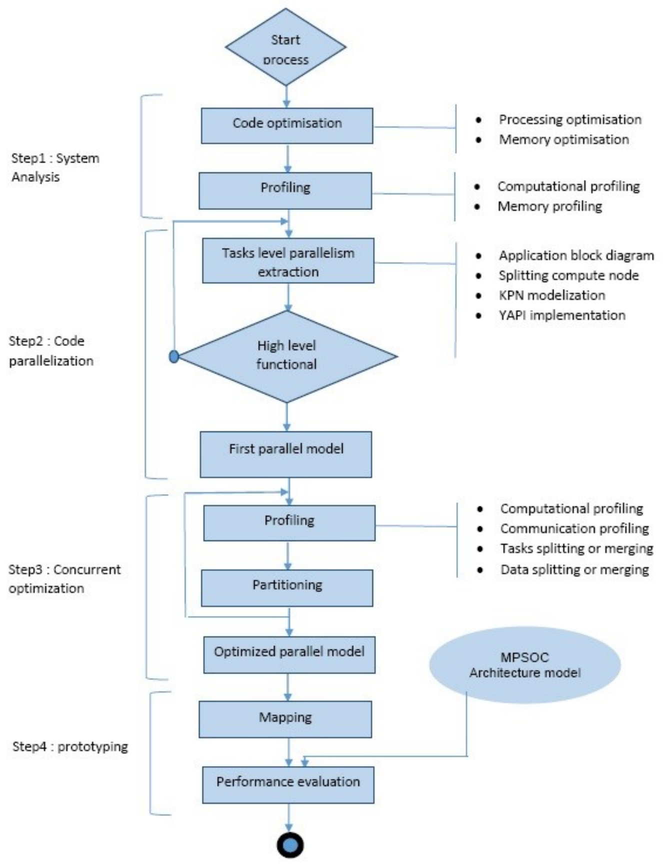


Figure 3 Parallelization approach Flowchart

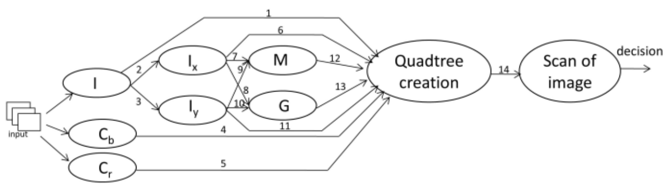
The output of this second step is an initial KPN model implemented through YAPI multi-threading programming interface. This obtained model is first validated using high level functional simulations. This consists to verifying that the parallel model carries out the same computation with the same functionalities as the original reference source code. The next step is "the concurrent optimization". This consists in balancing the communication and computation workload between the different processes of the original KPN/YAPI model defined in the previous step. For this aim, computation and communication profiling are performed to identify processes with high computing complexity and communication channels with unbalanced workload. Using communication and computation profiling results, different forms of task level splitting or merging and data level splitting are derived in a structured way to propose a model with more balanced computational workload and better communication behaviour. With task merging, channels transmitting large data values are removed. Data splitting is implemented for computational-expensive processes that are going to be split into multiple identical processes processing each a partition of the original data. For this case, a data dependency analysis is implemented before deciding for data partitioning to the spitted processes. The last step is "the real implementation". In this step the proposed model is implemented on a multi-core architecture.

4. Experimentation

A person matching application based on MSCOV is parallelized using the approach described above. In this section, a starting KPN/YAPI parallel model is first developed. The obtained performance results of the starting model are analyzed. For improved concurrency properties, an optimized parallel model is proposed. Finally the proposed model is validated using Zynq 7000 platform based on ARM cortex A9.

4.1 The initial parallel model

Starting from a C-code specification of a person matching application based on MSCOV, this code is first optimized in terms of memory usage and processing requirements. KPN/YAPI is then used to transform the optimized sequential C-code into a set of parallel communicating processes. The block diagram has served as a starting point to extract the maximum task level parallelism. KPN processes are extracted and inter-process communication is established using the message passing KPN primitives. The obtained KPN model is shown in Figure 4. This model acts as follows: First, the “I”, “Cr” and “Cb” processes collect image data from the input file. Then, "Ix" and "Iy" are computed using "I". After that, the "G" and "M" are computed using "Ix" and "Iy". The outputs of these processes represent the features needed as input for the "quadtree" process. When the quadtree structure is obtained, it will be forwarded to the "scan of image" process.



1	I_Quadtree creation		6	I_x_Quadtree creation		11	I_y_Quadtree creation
2	I_I_x		7	I_x_M		12	M_Quadtree creation
3	I_I_y		8	I_x_G		13	G_Quadtree creation
4	C_b_Quadtree creation		9	I_y_M		14	Quadtree creation_ Scan of image
5	C_r_Quadtree creation		10	I_y_G			

Figure 4 Initial parallel model of the MSCOV person matching technique

The KPN model of Figure 4 is implemented at the YAPI system level. The KPN/YAPI implementation is validated by high level functional simulation. The correctness of the parallel code is proved by comparing both execution results of sequential and parallel code using the same input. For performance evaluation of the proposed parallel model, two important functional properties are generated: the communication workload and the computation workload. The communication workload for a particular FIFO communication channel represents the amount of data exchange over this channel. The computation workload represents the processing time of each process in the network. These computation and communication characteristics define the concurrency properties of the model and measure the efficiency of the computation division over the different processes.

The communication workload

For the YAPI run-time environment the communication workload is defined as the number of tokens that are exchanged through the FIFOs channels. The obtained communication workload results of the starting model for an image of VGA resolution are shown in Figure 5.The Initial model has 13 FIFO channels transmitting 307200 tokens of 4 bytes size. So the total number of bytes communicated over each channel is equal to 1228800=307200*4 bytes. This big transmitted amount of data will require the involved processes to spend a lot of time dealing with communication. In addition, 3268590 (510*6409 bytes) are communicated over "quadtreecreaction_scan of image" channel. This makes the communication workload over a particular channel very high where as it remains comparatively low for the others. For a better communication balance, more data and task level splitting or merging techniques are required.

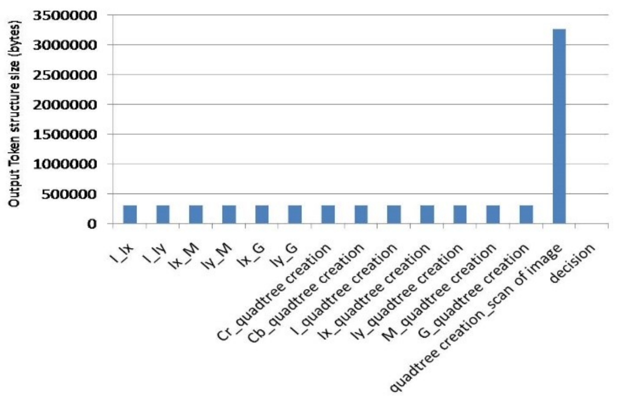


Figure 5 Communication workload of the Initial parallel model

Computation workload analysis

To better understand the concurrency properties of the starting parallel model, the computational workload in terms of CPU time percentage spent in the execution of each process is obtained in Figure 6.

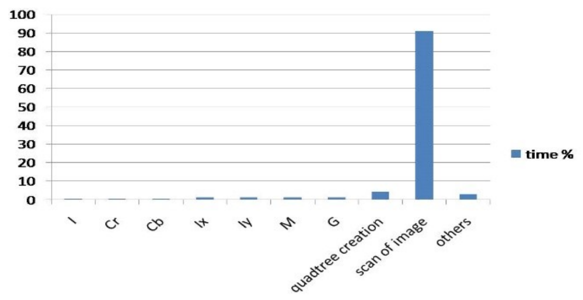


Figure 6 Parallel computational profiling of the first proposed model

This figure clearly shows that the Initial model has too much unbalanced computational workload. All processes have a negligible load compared to "scan of image" process which has the highest load. The "scan of image" process is very computational-expensive with more than 90% of the total computation time complexity.

Therefore, it is clear, using the obtained communication and computation workload results, that the initial parallel model of the person matching application based on MSCOV descriptor has poor concurrency properties. To get a better communication behavior, we should be using appropriate data level parallelism and task level splitting or merging. Data level parallelism consists in splitting the data exchanged over selected channels thus duplicating the associated processes of the model. Task level merging consists in merging the processes that are exchanging large data structures. Using task level splitting the available task-level parallelism is extracted by further splitting the computing nodes. The decision on using data splitting and task merging or splitting depends on the computational workload of the studied network. Generally, the processes, that have low computation with high communication loads, are merged while data splitting is applied for those with high computational workload.

4.2 The optimized parallel model

This section presents the different steps that have been used to derive in a structured way a parallel implementation of person matching application based on MSCOV with a balanced workload and good communication behavior. Using the profiling results of Figure 5 and Figure 6, it is clear that the processes "I", "Cb", "Cr", "Ix", "Iy", "M" and "G" have negligible computations loads and transmit quantitative data amounts to "quadtree creation" process. So we propose to merge all these processes into only one "feature_quadtree" process. In this case all features are calculated when the quadtree is created. So the associated channels transmitting the associated tokens structures are removed. On the other hand, the "scan of image" process is highly time consuming. This process execute the sliding mechanism that is carried out sequentially by processing one candidate region after the other as shown in Figure 7.

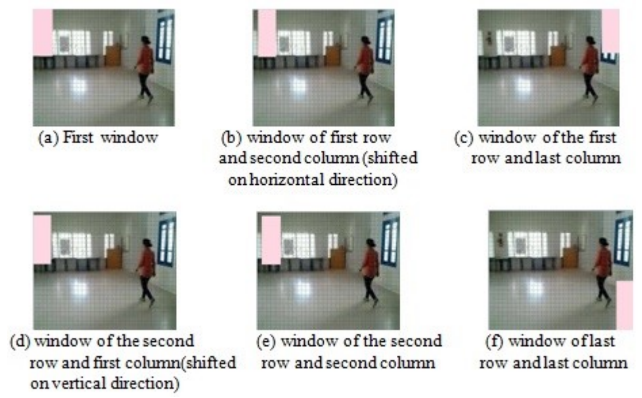
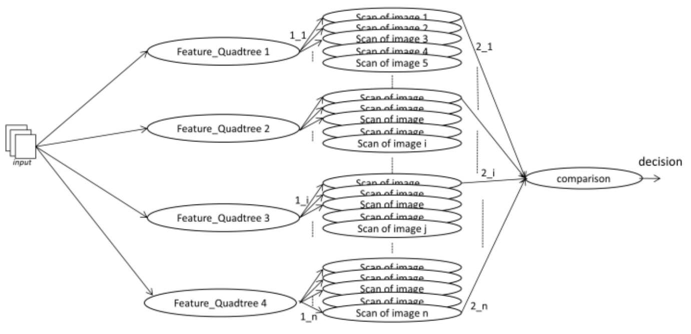

(a) First window
(b) window of first row and second column (shifted on horizontal direction)
(c) window of the first row and last column
(d) window of the second row and first column(shifted on vertical direction)
(e) window of the second row and second column
(f) window of last row and last column

Figure 7 Illustration of the scanning matching window

For a better concurrency optimization, the data splitting of the most computational-expensive "scan of image" process is proposed. The idea is to process overlapping windows of a single column in parallel. For this, a dedicated accumulation unit named a slice is introduced. Each slice calculates in parallel, the Euclidian distance between the obtained descriptors of the set of windows and the descriptor of the reference person. In fact, each process is allocated a set of scanning windows of the same column to be processed. After processing all columns, a "min distance" process should be introduced to extract the minimum distance of all scanning windows. The parallel model is shown in Figure 8.

Treating all scanning windows simultaneously reduce about 35 times the computation time of the "scan of image" process. However, a lot of communication workload (about 3268590 bytes) is exchanged between "feature_quadtree" and each process of scanning window. To reduce communication time, we suggest to duplicate quadtree representation into four sub quadtree processes. Theoretically, each image should be split into four parts each part has the dimensions (height x width/4). For each region a "feature_quadtree" process is performed. However before deciding for the partitioning, an analysis of data dependency should be considered to minimize data dependencies and maximize the parallelism rate between the four decomposed tasks.



1_1	Feature_ quadtree 1_Scan of image 1
1_2	Feature_ quadtree 1_Scan of image 2
1_i	Feature_ quadtree 3_Scan of image i
1_n	Feature_ quadtree 4_Scan of image n

2_1	Scan of image 1_comparison
2_2	Scan of image 2_comparison
2_i	Scan of image i_comparison
2_n	Scan of image n_comparison

Figure 8 Proposed optimized parallel KPN model of person matching application based on MSCOV descriptor

Data dependencies analysis

In this section, we are concerned with analyzing data dependencies of the "feature_quadtree" and "scan of image" processes as the data-partitioning is performed for these processes. The computation of covariance matrix of each region in the "scan of image" process, requires image characteristics presented on the quadtree structure. So each one requires about 3268590 bytes. Splitting "feature_quadtree" to four processes can decrease the communicated data per process to 823140 bytes. However regions within the cutting boundary will not be examined. This may effects the recognition result particularly if a person is in the cutting boundary. So we propose to split the image into four parts. In his connection parts 1, 2 and 3 has about (height x (with/4+ Hs-seuil)) resolution and part 4 has about (height x (with/4)) resolution. In this case the maximum workload communicated inter these processes in about 925650 bytes where scanning window size is 48x128 and the seuil of the scan is 16.

Concurrency results of the optimized model

The obtained parallel model of the person matching application based on MSCOV descriptor is given in Figure 8. This figure displays the task-merging of the "feature_quadtree" process, the data-partitioning for the "feature_quadtree" processes into "feature_quadtree1", "feature_quadtree2", "feature_quadtree3", and "feature_quadtree 4" processes and the data-partitioning for the "scan of image" process into the "scan of image 1", "scan of image 2", ..., "scan of image n" processes. This model has been implemented and validated by YAPI system level tool.

The communication and computational workload simulation results are shown respectively in Figure 9 and Figure 10 for VGA image resolution.

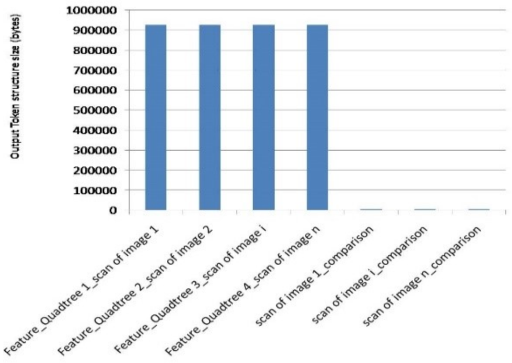


Figure 9 Communication workload of the optimized parallel model

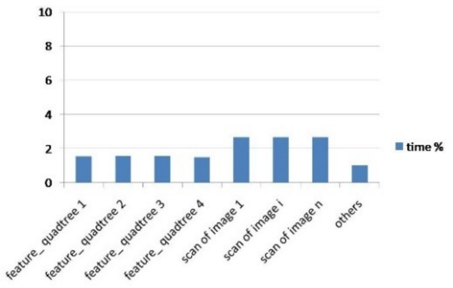
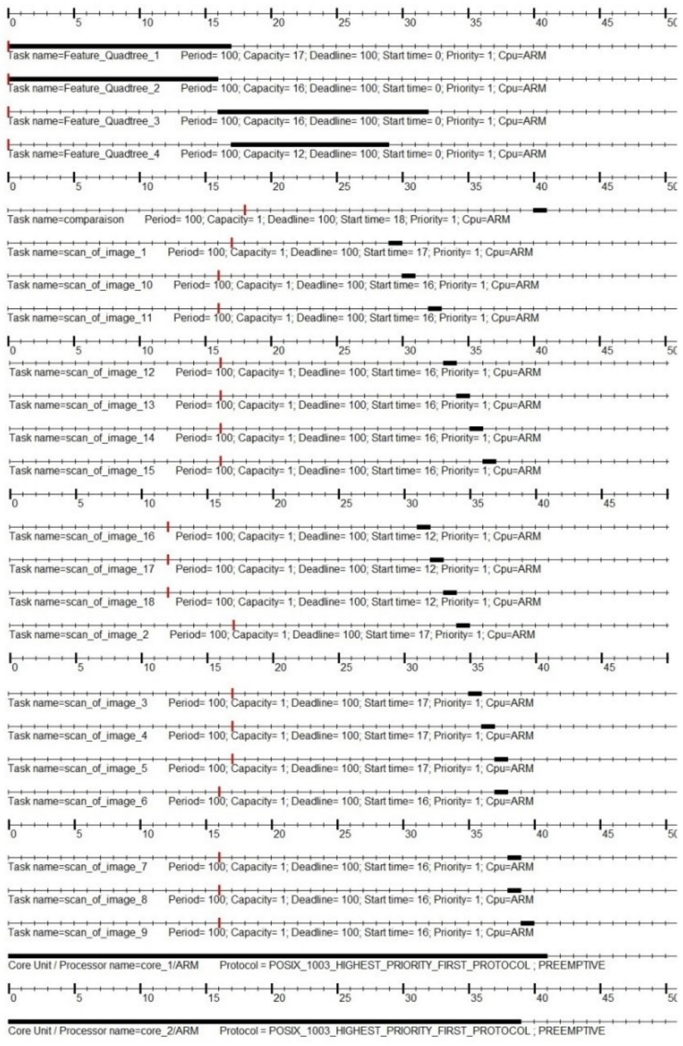


Figure 10 Parallel computational profiling of the final model

It is clear from Figure 9 that the optimized proposed model has better communication behavior compared to the initial model. The total number of communicated tokens are reduced. In addition, as indicated in Figure 10, merging tasks has decreased the time processes spent in communication and data spilling has distributed better the computation over processes. This final proposed model of the matching technique based on MSCOV descriptor has obviously better communication and computational behavior compared to the Initial model.

5 Implementation

The platform used to implement and validate the proposed parallel model is the Zynq®-7000 AP SoC ZC702 evaluation board. The Zynq-7000 family is ideal for embedded applications. It provides low power, high performance and ease of use [25]. Its average power consumption is about 1W while running at frequencies close to 1 GHz. It features a processing subsystem that implements a dual-hard core Cortex-A9 ARM supporting NEON technology. Considering all these advantages, Xilinx Zynq-7000 is chosen for prototyping the proposed solution of person matching application. The proposed model is obtained using a parallelization approach independent to the architecture. Therefore, prior to any real implementation, a quick prototyping of this task model is necessary to perform schedulability analysis and to determine which process will be executed on which processor. In this connection, many scheduling simulators and tools are proposed to model software architectures of real-time systems and to check its schedulability. CHEDDAR [26] is a free framework which provides tools to check if a real-time application meets its temporal constraints. The framework is based on the real-time scheduling theory to help for quick prototyping.



```
Task name=Feature_Quadtree_1    Period= 100; Capacity= 17; Deadline= 100; Start time= 0; Priority= 1; Cpu=ARM
Task name=Feature_Quadtree_2    Period= 100; Capacity= 16; Deadline= 100; Start time= 0; Priority= 1; Cpu=ARM
Task name=Feature_Quadtree_3    Period= 100; Capacity= 16; Deadline= 100; Start time= 0; Priority= 1; Cpu=ARM
Task name=Feature_Quadtree_4    Period= 100; Capacity= 12; Deadline= 100; Start time= 0; Priority= 1; Cpu=ARM
Task name=comparaison    Period= 100; Capacity= 1; Deadline= 100; Start time= 18; Priority= 1; Cpu=ARM
Task name=scan_of_image_1    Period= 100; Capacity= 1; Deadline= 100; Start time= 17; Priority= 1; Cpu=ARM
Task name=scan_of_image_10    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_11    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_12    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_13    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_14    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_15    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_16    Period= 100; Capacity= 1; Deadline= 100; Start time= 12; Priority= 1; Cpu=ARM
Task name=scan_of_image_17    Period= 100; Capacity= 1; Deadline= 100; Start time= 12; Priority= 1; Cpu=ARM
Task name=scan_of_image_18    Period= 100; Capacity= 1; Deadline= 100; Start time= 12; Priority= 1; Cpu=ARM
Task name=scan_of_image_2    Period= 100; Capacity= 1; Deadline= 100; Start time= 17; Priority= 1; Cpu=ARM
Task name=scan_of_image_3    Period= 100; Capacity= 1; Deadline= 100; Start time= 17; Priority= 1; Cpu=ARM
Task name=scan_of_image_4    Period= 100; Capacity= 1; Deadline= 100; Start time= 17; Priority= 1; Cpu=ARM
Task name=scan_of_image_5    Period= 100; Capacity= 1; Deadline= 100; Start time= 17; Priority= 1; Cpu=ARM
Task name=scan_of_image_6    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_7    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_8    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Task name=scan_of_image_9    Period= 100; Capacity= 1; Deadline= 100; Start time= 16; Priority= 1; Cpu=ARM
Core Unit / Processor name=core_1/ARM    Protocol = POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL ; PREEMPTIVE
Core Unit / Processor name=core_2/ARM    Protocol = POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL ; PREEMPTIVE
```

Figure 11 Scheduling simulation of proposed model of QVGA image

5.1 Scheduling simulation

Figure 11 is a screenshot of CHEDDAR. This scheduling is drawn for the proposed model targeting dual core architecture according to the preemptive policy SCHED_RR using the POSIX 1003.1b scheduler. According to Figure 11, each process of the proposed model is running to the end without releasing the processor.

Processes are mapped as follow: "feature_quadtree 1", "feature_quadtree 3" and their corresponding "scan of image i" are mapped into core 1 and "feature_quadtree 2", "feature_quadtree 4" and their corresponding "scan of image i" are mapped into core 2. The final process "comparison" is mapped in core 1.

5.2 Parallel implementation

An example of matching technique by MSCOV descriptor using QVGA image resolution is illustrated on Figure 12. The execution time is illustrated on Table 1 using the two versions of implementation: original sequential implementation on 1 core ARM cortex-A9 and parallel implementation of proposed model on 2 cores ARM cortex-A9.

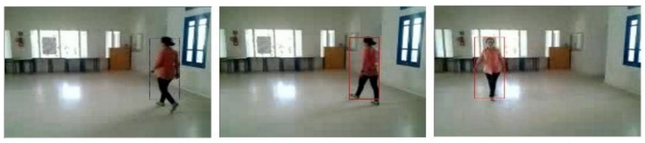


Figure 12 Example of matching person based on MSCOV descriptor

As shown in Table 1, the execution time of the person matching application based on MSCOV descriptor using the proposed model implemented on dual cores ARM cortex A9 is 1.9 x faster than sequential implementation on mono-core. Also, the number of frame per second is increased to achieve about 16.33 fps using dual core.

Table 1. Execution time for serial and parallel implementation

	1 core ARM cortex A9	*2 cores ARM cortex A9*
Execution time	116.49 ms	61.21ms
Fps	8.58 fps	16.33 fps

6 Conclusion

This paper presented an optimized parallel model of a person matching technique based on MSCOV descriptor, developed using a high-level independent target-architecture parallelization approach. This approach is based on the use of KPN parallel programming models. It is characterized by the exploration of task and data levels of parallelism and merging to ensure the optimal balance of the model computation and communication behavior. In this context, we proposed an initial parallel model that extracts the maximum task level parallelism. This model is then implemented and validated using the YAPI environment. The communication and computation workload analysis of the KPN/YAPI initial model has showed very poor concurrency properties. To improve the model concurrency properties, data level parallelism and task level splitting or merging are applied. At the end, an optimized parallel model of the matching technique based on MSCOV descriptor is obtained. This model gives considerable computation and inters process communication workload balance without sacrificing detection quality. This model is validated by a real implementation on dual core architecture. The proposed model has proven to be effective since it reachs 16.33 fps in a dual core implementation. We are planning to perform further studies on maximizing the power of ARMs with DSP/GPU accelerators.

References

[1] R. Lienhart, J. Maydt (2002), An extended set of Haar-like features for rapid object detection, International conference on image processing, pp. 900–903.

[2] G. David (2004), “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, vol. 60, pp. 91–110.

[3] N. Dalal, B. Triggs, “Histograms of oriented gradients for human detection,” IEEE computer society. computer vision and pattern recognition, vol. 1, pp.

[4] O. Tuzel, F. Porikli, P. Meer (2008), “Pedestrian detection via classification on riemannian manifolds,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, pp. 1713-1727.

[5] J.M. Michael, W. Marcel, W.M.S. Arnold (2010), “Color based tracing in real-life surveillance data,” Transactions on Data Hiding and Multimedia Security. Springer-Verlag, Berlin, Heidelberg, vol. 6010, pp 18-33.

[6] W. Ayedi, H. Snoussi, M. Abid (2011), “A fast multi-scale covariance descriptor for object re-identification,” Pattern Recognition Letters, vol. 33, pp. 1902–1907.

[7] W. Ayedi, H. Snoussi, F. Smach, M. Abid (2012), “The multi-scale covariance descriptor: Performances analysis in human detection,” Biometric Measurements and Systems for Security and Medical Applications, IEEE Workshop, pp. 1-5.

[8] W. Ayedi, H. Snoussi, F. Smach, M. Abid (2012), “Tree based object matching using multi-scale covariance descriptor,” Proc. International Conference on Image Processing, Computer Vision, and Pattern Recognition, 2012. (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[9] M. Bramberger, A. Doblander, A. Maier, B. Rinner, H. Schwabach (2006), “Distributed embedded smart cameras for surveillance applications,” Computer, vol. 39, no. 2, pp. 68–75.

[10] A. Kerhet, M. Magno, F. Leonardi, A. Boni, and L. Benini (2007), “A low-power wireless video sensor node for distributed object detection,” Journal of Real-Time Image Processing, vol. 2, pp. 331–342.

[11] Y. Yang, C. Chiu (2014), "Boosted multi-class object detection with parallel hardware implementation for real-time applications," in Acoustics, Speech and Signal Processing , IEEE International Conference, pp. 7530–7534.

[12] A. Suleiman, V. Sze (2016), "An energy-efficient hardware implementation of hog-based object detection at 1080hd 60 fps with multi-scale support," Journal of Signal Processing Systems, vol. 84, pp. 325-337.

[13] G. Kahn (1974), "The semantics of a simple language for parallel programming," Proceedings of IFIP 74, North Holland

[14] E. Kock, G. Essink, W. Smits (2000), P.Wolf, J.Y. Brunel, W.M Kruijtzer, P. Lieverse, K.A. Vissers, "YAPI:Application modeling for signal processing system," IEEE Procceedings of the 37th Annual Design Automation Conference, pp. 402-405.

[15] Y. Said, M. Atri (2016), "Efficient and high-performance pedestrian detector implementation for intelligent vehicles," IET Intelligent Transport Systems, vol. 10, pp. 438-444.

[16] J. Wang, K. Yan, K. Guo, J. Yu, L. Sui, S. Yao, Y. Wang (2016), "Real-time pedestrian detection and tracking on customized hardware," 14th Symposium on Embedded Systems For Real-time Multimedia.

[17] P. Y. Hsiao, S.Y. Lin, C. Y. Chen (2016), "A real-time fpga based human detector," IEEE International Symposium on Computer, Consumer and Control, pp. 1014-1017.

[18] Y. Zhu, Y. Liu, D. Zhang, S. Li, P. Zhang (2010), "Acceleration of pedestrian detection algorithm in novel c2rtl hw/sw codesign platform," International Conference Green Circuits and Systems, pp 615 - 620,

[19] H. Mao, M. Takaaki, A. Hideharu (2015), "Data reduction and parallelisation for human detection system," The 19th Workshop on Synthesis And System Integration of Mixed Information Technologies, pp. 134 - 139

[20] P. Y. Hsiao, S.Y. Lin , S. Huang (2015), "An FPGA based human detection system with embedded platform," Microelectronic Engineering, vol. 138, pp. 42–46.

[21] P. Y. Chen, C. C. Huang, C. Y. Lien, and Y. H. Tsai, "An efficient, hardware implementation of hog feature extraction for human detection," IEEE Transactions on Intelligent Transportation Systems., vol. 15 , pp. 656 - 662, 2013.

[22] S. Martelli, D. Tosato, M. Cristani (2011), "Fast FPGA-based architecture for pedestrian detection based on covariance matrices," IEEE International Conference on Image Processing, pp. 389 –392.

[23] A. Nesrine, A. Walid, A.C. Ammari, A. Mohamed (2014), "SW/HW implementation of image covariance descriptor for person detection system," Advanced Technologies for Signal and Image Processing (ATSIP), 2014 1st International Conference on, pp. 115 – 119.

[24] J. Chhugani , W. Macy , A. Baransi , D. Nguyen , M. Hagog , S. Kumar , W. Lee , Y. Chen , P. Dubey (2008), "Efficient Implementation of Sorting on Multi-Core SIMD CPU Architecture," Proceedings of the VLDB Endowment, vol. 1, pp. 1313-1324.

[25] Xilinx Inc. UG585, "Zynq-7000 extensible processing platform technical reference manual," 2012.

[26] The Cheddar project : a GPL real-time scheduling analyzer, http://beru.univ-brest.fr/~singhoff/cheddar/

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

National Ada Organizations

Ada-Belgium

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

Ada in Denmark

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

Ada-Deutschland

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

Ada-France

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

Ada-Spain

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

Ada-Switzerland

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