ADA USER JOURNAL

Volume 35
Number 3
September 2014

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

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

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

Proceedings and Special Issues

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

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

News and Product Announcements

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

Reprinted Articles

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

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

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

Editorial

This September issue of the Ada User Journal continues the publication of contributions which originate from the Ada-Europe 2014 conference. First, the reader will find a paper from the industrial track of the conference, authored by Ana Rodríguez from Silver Atena, Spain, and that discusses the development of critical software in the scope of the first European rail traffic management system. Afterwards, Jacob Sparre Andersen, from JSA Research & Innovation, Denmark, presents an experience report on when Java leaks privacy through the getter methods.

The third technical contribution in this issue is a paper, by André Pedro, from the CISTER Research Centre, Portugal, that presents an ongoing work on supporting Ada Task types in the AVR-Ada compiler for 8-bit AVR platforms. The technical content of the issue finalizes with the continuation of the SPARK 2014 Rationale series. In this paper, Pavlos Efstathopoulos, of Altran, UK, presents two contributions on the topics of data dependencies and information flow.

The reader will also find in this issue the usual News Digest, Calendar and Forthcoming Events sections, provided by the respective editors. In particular, the forthcoming events section provides information about the program of the upcoming SIGAda High Integrity Language Technology Conference, which will take place next October in Portland, Oregon, and call for contributions for ARCS 2015, the GI/ITG International Conference on Architecture of Computing Systems, which will take place in Porto, Portugal, March 2015; for IRTAW 2015, the International Real-Time Ada Workshop – a very welcomed return – which will be held in Vermont, USA, April 2015; and our flagship conference, Ada-Europe 2015, which will be hosted by the Universidad Politécnica de Madrid, Spain.

Luís Miguel Pinho
Porto
September 2014
Email: AUJ_Editor@Ada-Europe.org

Quarterly News Digest

Jacob Sparre Andersen

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

Contents

Ada-related Events

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

Ada-Belgium Spring Event

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Tue, 3 Jun 2014 22:34:58 +0000
Subject: Ada-Belgium Spring 2014 Event, Sun 15 June 2014
Newsgroups: comp.lang.ada, fr.comp.lang.ada, be.comp.programming

---

Ada-Belgium Spring 2014 Event

Sunday, June 15, 2014, 12:00-19:00

Wavre area, south of Brussels, Belgium

including at 15:00

2014 Ada-Belgium General Assembly

and at 16:00

Ada Round-Table Discussion

<http://www.cs.kuleuven.be/~dirk/ada-belgium/events/local.html>

Announcement

The next Ada-Belgium event will take place on Sunday, June 15, 2014 in the Wavre area, south of Brussels.

For the 7th year in a row, Ada-Belgium decided to organize their "Spring Event", which starts at noon, runs until 7pm, and includes an informal lunch, a key signing party, the 21st General Assembly of the organization, and a round-table discussion on Ada-related topics the participants would like to bring up. Afterwards, those interested can once more get practical hands-on experience on packaging Ada software for Debian with Ludovic Brenta, principal maintainer of Ada in Debian.

Schedule

- 12:00 welcome and getting started (please be there!)

- 12:15 informal lunch

- 14:45 key signing party

- 15:00 Ada-Belgium General Assembly

- 16:00 Ada round-table + informal discussions

- 19:00 end

Participation

Everyone interested (members and non-members alike) is welcome at any or all parts of this event.

For practical reasons registration is required. If you would like to attend, please send an email before Wednesday, June 11, 21:00, to Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be> with the subject "Ada-Belgium Spring 2014 Event", so you can get precise directions to the place of the meeting. Even if you already responded to the preliminary announcement, please reconfirm your participation ASAP.

If you are interested to become a new member, please register by filling out the 2014 membership application form[1] and by paying the appropriate fee before the General Assembly. After payment you will receive a receipt from our treasurer and you are considered a member of the organization for the year 2014 with all member benefits[2]. Early renewal ensures you receive the full Ada-Belgium membership benefits (including the Ada-Europe indirect membership benefits package).

As mentioned at earlier occasions, we have a limited stock of documentation sets and Ada related CD-ROMs that were distributed at previous events, as well as back issues of the Ada User Journal[3]. These will be available on a first-come first-serve basis at the General Assembly for current and new members. (Please indicate in the above-mentioned registration e-mail that you're interested, so we can bring enough copies.)

[1] http://www.cs.kuleuven.be/~dirk/ada-belgium/forms/member-form14.html

[2] http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html

[3] http://www.ada-europe.org/auj/home/

Informal lunch

The organization will provide food and beverage to all Ada-Belgium members. Non-members who want to participate at the lunch are also welcome: they can choose to join the organization or pay the sum of 15 Euros per person to the Treasurer of the organization.

General Assembly

All Ada-Belgium members have a vote at the General Assembly, can add items to the agenda, and can be a candidate for a position on the Board[4]. See the separate official convocation[5] for all details.

[4] http://www.cs.kuleuven.be/~dirk/ada-belgium/board/

[5] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/14/140615-abga-conv.html

Key Signing Party

Wouldn't it be nice if a majority of people used GPG to sign their email every day so that you could send all non-signed email into the spam bin? To make that dream come true, please join and expand the global Web of Trust[6]!

What you should bring with you:

- an official ID card issued by your national government;

- your GPG key fingerprint (i.e. the output of gpg --fingerprint) on small paper slips; a dozen copies or so should be enough.

What you will go home with:

- signatures from all other participants;

- automatic inclusion in the global Web of Trust;

- the ability to digitally sign or encrypt anything you like.

[6] http://en.wikipedia.org/wiki/Web_of_Trust

Ada Round-Table Discussion

As last year, we plan to keep the technical part of the Spring event informal as well. We will have a round-table discussion on Ada-related topics the participants would like to bring up. We invite everyone to briefly mention how they are using Ada in their work or non-work environment, and/or what kind of Ada-related activities they would like to embark on. We hope this might spark some concrete ideas for new activities and collaborations.

Afterwards, those interested can get practical information and hands-on experience on "Packaging Ada Software for Debian"[7][8].

[7] http://www.debian.org/

[8] http://people.debian.org/~lbrenta/debian-ada-policy.html

Directions

To permit this more interactive and social format, the event takes place at private premises in the Wavre area, south of Brussels. As instructed above, please inform us by e-mail if you would like to attend, and we'll provide you precise directions to the place of the meeting. Obviously, the number of participants we can accommodate is not unlimited, so don't delay...

Looking forward to meet many of you!

Dirk Craeynest, President Ada-Belgium

Dirk.Craeynest@cs.kuleuven.be

Acknowledgements

We would like to thank our sponsors for their continued support of our activities: AdaCore, Barco, Katholieke Universiteit Leuven (KU Leuven), and Université Libre de Bruxelles (U.L.B.).

If you would also like to support Ada-Belgium, find out about the extra Ada-Belgium sponsorship benefits:

http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html#sponsor

Ada-related Resources

Pre-Ada Documents

From: Pascal Obry <pascal@obry.net>
Date: Fri, 23 May 2014 17:41:41 +0200
Subject: Pre-Ada documents - Green Language
Newsgroups: comp.lang.ada

You may be interested by those PDF:

http://pobry.blogspot.fr/2014/05/pre-ada-documents.html

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Fri, 23 May 2014 16:51:47 -0700
Subject: Re: Pre-Ada documents - Green Language
Newsgroups: comp.lang.ada

> [...]

The Red RM is at

http://www.iment.com/maida/computer/redref/index.htm

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 23 May 2014 16:52:16 -0500
Subject: Re: Pre-Ada documents - Green Language
Newsgroups: comp.lang.ada

> [...] I'd be very interested in Red and/or Blue if you happen to come across them.

You have some aversion to Yellow? :-)

(There were four original proposals. And I think they all were published in some ACM publication (SIGPlan Notices??) - I remember reading them back when I was still a student. There's only a detailed proposal for Red and Green, of course, Yellow and Blue were chopped early.

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Tue, 27 May 2014 16:06:02 -0400
Subject: Re: Pre-Ada documents - Green Language
Newsgroups: comp.lang.ada

> [...] from what I heard [...] it was basically just Pascal with a few alterations/additions.

All four were supposedly based on Pascal. I believe Yellow was closely based on Pascal, but still, more than "basically just Pascal with...", and the others were much more loosely based on Pascal.

Ada (even Ada 83) is of course hugely different from Pascal.

From: Luke A. Guest <laguest@archeia.com>
Date: Tue, 27 May 2014 16:15:49 +0000
Subject: Re: Pre-Ada documents - Green Language
Newsgroups: comp.lang.ada

I would like to see the blue one specifically as I hear it was particularly weird.

Repositories of Open Source Software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Thu Jul 31 2014
Subject: Repositories of Open Source software

AdaForge: 8 repositories [1]

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

Codelabs: 20+ repositories [3]

GitHub: 581 repositories [4]
126 developers [5]

Rosetta Code: 604 examples [6]
28 developers [7]

Sourceforge: 239 repositories [8]

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

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

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

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

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

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

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

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

[See also “Repositories of Open Source Software”, AUJ 35-2, p. 74. —sparre]

Ada-related Tools

GWindows Setup

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Wed, 9 Apr 2014 05:16:49 -0700
Subject: Ann: GWindows Setup 5-Apr-2014
Newsgroups: comp.lang.ada

An update of the GWindows framework is packaged in a standalone installer.

---> GWindows Setup 5-Apr-2014.exe

It can be downloaded at http://sf.net/projects/gnavi/

NB: the GNATCOM framework is included as well.

Major changes in the framework since the last announce here - numbers below refer to svn repository revisions:

228: Added GWindows.Menus.Immediate_Popup_Menu

225: Added GWindows.Taskbar and Test_Taskbar

216: Added GWindows.System_Tray and Test_System_Tray

213: Added GWindows.Locales and Test_Locales

206: GWindows.Common_Controls: added Item_At_Position method for Tree_View_Control_Type

205: Beginning of a new tutorial #24 about Drag & Drop

[See also “GWindows Setup”, AUJ 34-1, p. 8. —sparre]

AdaControl

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 10 Apr 2014 17:11:45 +0200
Subject: [Ann] AdaControl 1.16r10 released
Newsgroups: comp.lang.ada

Adalog is pleased to announce a new release of AdaControl, featuring 4 brand new rules and plenty of new subrules and improvements, for a whopping total of 513 possible checks!

As always, the latest version is available from

http://www.adalog.fr/adacontrol2.htm or from Sourceforge.

AdaControl is free software, and can be used and modified without restrictions.

Adalog provides commercial support for AdaControl, as well as consultancy in using the tool for special purposes and developing coding standards.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 05 Jun 2014 17:32:08 +0200
Subject: Adacontrol 1.16r11 released
Newsgroups: comp.lang.ada

This release fixes a bug, cleans up minor things, but the main change is that the executable versions are now provided for GNAT-GPL-2014.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Mon, 16 Jun 2014 12:04:43 +0200
Subject: Adacontrol: change of BT system
Newsgroups: comp.lang.ada

AdaControl now uses the normal "Ticket" system of SourceForge for improvement suggestions, bug reports, etc. Simply go to the Adalog page (http://sourceforge.net/projects/adacontrol) and click on "Ticket".

Submitting tickets does not require SourceForge identification, although our supported users are invited to use their AdaControl/SourceForge account for priority treatment of their requests.

The old system (MantisBT) is being discontinued.

[See also “AdaControl”, AUJ 34-3, p. 140. —sparre]

PragmAda Reusable Components

From: PragmAda Software Engineering <pragmada@pragmada.x10hosting.com>
Date: Sat, 12 Apr 2014 10:14:02 -0700
Subject: ANN: Beta PragmARCs for ISO/IEC 8652:2007
Newsgroups: comp.lang.ada

A new version of the beta release of the PragmAda Reusable Components for ISO/IEC 8652:2007 is available at

http://pragmada.x10hosting.com/pragmarc.htm

This version includes an implementation of the Threefry random-number generator, an Unbounded_Integers package, and a Rational_Numbers package.

Threefry is a fully CRUSH compliant random-number generator, and is claimed to be the fastest such generator. Threefry is an encryption-based generator, derived from the Threefish encryption scheme. This implementation assumes the existence of Interfaces.Unsigned_64.

Unbounded_Integers implements integers bounded only by available memory. Many similar packages implement bounded integers, such as the package that is part of the Ada Crypto library. This implementation uses a standard container to avoid explicit access types and memory management.

Rational_Numbers uses Unbounded_Integers to implement unbounded rational values.

Error reports, comments, and suggestions are always welcome.

[See also “PragmAda Reusable Components”, AUJ 34-2, p. 66. —sparre]

GNAT Runtimes for Microprocessors

From: Brian Drummond <brian@shapes.demon.co.uk>
Date: Mon, 26 May 2014 08:26:20 GMT
Subject: Re: Where to get Zero Footprint Profile?
Newsgroups: comp.lang.ada

> [...]

There is a slightly more developed version for the AVR microprocessors, which appears to support a secondary stack [1].

I found it relatively easy to adapt for the TI MSP430 processors [2].

You may get some idea of what's involved by comparing these two, and adapting as required for your target processor. If so, please put your work out there so that someday, we may draw together all these disparate efforts into something more coherent...

Another unrelated effort targeting ARM processors [exists, with the intent eventually to support the Ravenscar profile [3].

[1] http://sourceforge.net/projects/avr-ada/

[2] http://sourceforge.net/projects/msp430ada/

[3] http://sourceforge.net/projects/arm-ada/

Tables

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 2 Jun 2014 19:36:07 +0200
Subject: Tables for Ada v1.12
Newsgroups: comp.lang.ada

The library provides tables searched using string keys. The binary search is used for names of known length. It is also possible to search a table for names of unknown length, i.e. to parse a string using the table. In this case the search time is near to logarithmic, but in the worst case can be linear (when the table contains tokens like "a", "aa", "aaa" and so on).

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

The new version is compiled with GNAT 4.9.

Ada-Fuse

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 3 Jun 2014 12:00:08 +0200
Subject: Ada-Fuse, status of
Newsgroups: comp.lang.ada

This one:

https://github.com/RanaExMachina/ada-fuse

It looks quite interesting. What is the status of. Is it maintained?

From: Stefan Lucks <stefan.lucks@uni-weimar.de>
Date: Fri, 13 Jun 2014 12:31:23 +0200
Subject: Re: Ada-Fuse, status of
Newsgroups: comp.lang.ada

> [...]

It has been written by students of mine, who currently are not interested in maintaining it.

Strings_Edit

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 4 Jun 2014 22:09:55 +0200
Subject: ANN: Strings edit for Ada v 2.9
Newsgroups: comp.lang.ada

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

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

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

Changes to the version 2.8:

- Added wildcard matching with character mapping equivalence;
- The package Strings_Edit.UTF8.Wildcards.Case_Insensitive provides case-insensitive wildcard matching;
- Compiled with GNAT 4.9.

[See also “Strings_Edit library”, AUJ 33-2, p. 77. —sparre]

Simple Components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 5 Jun 2014 18:39:50 +0200
Subject: ANN: Simple components v 4.0 released
Newsgroups: comp.lang.ada

The library 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 designing tools.

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

Changes to the version 3.22:

- ODBC bindings bug causing a connection left opened despite object finalization was fixed;
- Generic_Blackboard supports GCC platforms with no Pragma Atomic available for 64-bit integers;
- SQLite3 bindings are switched to the amalgamation version 3080200;
- Column_Type and Is_Valid functions added to the package SQLite;
- HTTP SQLite3 database browser added;
- Blocking file access support added;
- Persistent memory pools added;
- B-tree implementation added.

[See also “Simple components”, AUJ 34-2, p. 66. —sparre]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 26 Jul 2014 10:39:30 +0200
Subject: ANN: Simple components for Ada v4.1
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 designing tools. It grew out of needs and does not pretend to be universal.

Tables management and strings editing are described in separate documents see Tables and Strings edit. The library is kept conform to the Ada 95, Ada 2005, Ada 2012 language standards.

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

The release fixes some minor bugs.

GtkAda Contributions

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 7 Jun 2014 18:18:08 +0200
Subject: ANN: GtkAda contributions v3.8
Newsgroups: comp.lang.ada

The library provides various additions to GtkAda bindings to GTK+.

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

Changes to the version 2.14:

- The library was adapted to the GtkAda 3.x. Earlier versions are no more supported;
- Only Ada 2005 and Ada 2012 are supported when GtkAda 3.x is used;
- Null address exclusion is added where appropriate (since Ada 95 is no more supported);
- Add_Class_Style, Add_Widget_Class_Style, Add_Widget_Name_Style removed from Gtk.Missed. The resource files are replaced with GTK+ style provider;
- Set_Has_Tooltip is removed from Gtk.Missed. It is now provided by Gtk.Widget;
- Has_Tooltip is removed from Gtk.Missed. It is now provided by Gtk.Widget as Get_Has_Tooltip;
- The signature of Set_Tip from Gtk.Missed is changed. The first parameter is removed;
- Get_Size is removed from Gtk.Missed. It is now provided by Gtk.Window;
- Delete_Event_Handler and Destroy_Handler added to Gtk.Missed to ease designing GtkAda applications consisting of a single procedure;
- Init of Gtk.Main.Router has additional parameter, the application window. The change is necessary because GTK+ 3.x does not offer hooks on main loop exit as GTK+ 2.x did;
- Gtk.Object.Checked_Destroy is renamed to GLib.Object.Checked_Destroy;
- Gtk.Generic_Style_Button handling tooltips is changed to the new interface;
- Class_Find_Style_Property removed from Gtk.Widget.Styles. See Gtk.Widget.Find_Style_Property;
- Get_Property for GFloat removed from Gtk.Missed as it now provided by the package GLib.Properties;
- Class_Install_Property added to Gtk.Missed;
- From_RGBA and To_RGBA added to Gtk.Missed;
- Procedure Check and function To_String were added to Gtk.Missed;
- Get_Screen_Position procedure added to Gtk.Missed;
- Generic package Set_Column_Cell_Data added to Gtk.Missed (sets cell data function of a tree model column);
- Set_Object removed from GLib.Values.Handling. This procedure is now provided in GLib.Values;
- The package Gtk.Widget.Styles.CSS_Store added to enumerate and export widget style properties in the CSS format;
- The package Gtk.Tooltips.Strong_References was removed;
- The package Gtk.Cell_Renderer.Abstract_Renderer was adapted to new interface of cell renderer introduced by GTK+ 3.x;
- The package Gtk.Tree_Model.Abstract_Store was adapted to new interface of cell renderer introduced by GTK+ 3.x;
- Procedures Forward_Search and Backward_Search were removed from the package Gtk.Source_Buffer. This functionality is now provided by Gtk_Text_Iter;
- Package Gtk.Source_Mark_Attributes added (new in GtkSourceView 3.x);
- The procedures Get_Mark_Category_*, Set_Mark_Category_* were removed from the package Gtk.Source_View. This functionality is provided by Gtk.Source_Mark_Attributes in GtkSourceView 3.x;
- Procedures Get_For_Screen, Get_Limit, Set_Limit were removed from Gtk.Recent_Manager_Alt, as they are no more present in GTK+ 3.x;
- Package Gdk.Pixbuf.Image was added to provide memory-mapped images which pixels can be manipulated directly in the memory;
- Get_Style_Fallback added to the package Gtk.Source_Language.

[See also “GtkAda Contributions”, AUJ 33-3, p. 145. —sparre]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 27 Jul 2014 09:41:49 +0200
Subject: ANN: GtkAda contributions v3.9 released
Newsgroups: comp.lang.ada

The library is a contribution to GtkAda, an Ada bindings to GTK+ toolkit. It deals with the following issues: tasking support; custom models for tree view widget; custom cell renderers for tree view widget; multi-columned derived model; an extension derived model (to add columns to an existing model); an abstract caching model for directory-like data; tree view and list view widgets for navigational browsing of abstract caching

models; file system navigation widgets with wildcard filtering; resource styles; capturing the resources of a widget; embeddable images; some missing sub-programs and bugfixes; a measurement unit selection widget and dialogs; an improved hue-luminance-saturation color model; simplified image buttons and buttons customizable by style properties; controlled Ada types for GTK+ strong and weak references; and a simplified means to create lists of strings.

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

Changes to the previous version:

- Minor bug fixes;
- Fedora packages rely on the official GtkAda3 packages (released by Björn Persson).

Units of Measurement

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 9 Jun 2014 17:16:59 +0200
Subject: ANN: Units of measurement for Ada v 3.4 released
Newsgroups: comp.lang.ada

The library provides means to handle dimensioned values in Ada. It also includes GTK+ widgets and renderers for unit selection and rendering dimensioned values.

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

Changes to the version 3.3:

- The widgets and renderers are adapted for the GTK+ 3.x. GTK+ 2.x is no more supported;
- Procedure Split is added to the package Units;
- Only Ada 2005 and Ada 2012 are supported when widgets and renderers are used. The non-GUI parts of the software remain Ada 95 conform;
- Bug fix in text conversion that led to false output of values with units like square meter;
- Compiled with GNAT 4.9.

[See also “Units of Measurement”, AUJ 34-2, p. 68. —sparre]

Wish-list

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue, 10 Jun 2014 22:34:12 +0300
Subject: Re: a new language, designed for safety !
Newsgroups: comp.lang.ada

I would like to have following non-GPL, preferably 100% Ada, libraries without GNAT specific dependencies:

- GUI library
- JSON library (well, I have written my own, but it isn't perfect yet)
- Tiny-YAML library [1]
- HTTP client library with SSL (using curl bindings for now)
- Plain socket library with poll/epoll support
- Some sort of easy interface to execute programs on *nix/Windows and capture the input/output/stderr/exit code[2]
- Server side web framework which does not require the latest GNAT to work (=works with older GNAT releases and other compilers also)
- At least semi-decent runtime/peripheral library for ARM Cortex-Mx processors[3]
- Interface to sqlite databases (using my own partial bindings atm.)
- reStructuredText to HTML formatter
- Antlr4 runtime

For many things bindings to existing C libraries would be fine, but usually authors select different licenses for Ada bindings than what the original C library has. Like why on earth use plain GPL or even GMGPL for bindings if original C library is distributed under public domain, MIT, or BSD license?!

Of course, authors can do that, but it still baffles me and causes some extra headache when I need to decipher can I combine the licenses and the code with my own code.

[1] http://search.cpan.org/dist/YAML-Tiny/lib/YAML/Tiny.pm

[2] *nix part is easy, but I have no idea how to do that on Windows :)

[3] Like libopencm3, http://libopencm3.org, or http://mbed.org

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 11 Jun 2014 10:45:44 +0200
Subject: Re: a new language, designed for safety !
Newsgroups: comp.lang.ada

> [...]

> - GUI library

There is not much choice, since the C GUI libraries aren't GM GPL either.

[...]

> - Plain socket library with poll/epoll support

Adasockets? I didn't use it for a long time, however. The last time I looked at, it had a makefile, which was a non-starter to me. But I think it is possible to make a decent portable high-level socket library for Windows/Linux/VxWorks, if there were interest.

> - Some sort of easy interface to execute programs on *nix/Windows and capture the input/output/stderr/exit code[2]

I am using it from GTK+. It does all that under both Windows and Linux, even capturing output into a text buffer:

http://www.dmitry-kazakov.de/ada/gtkada_contributions.htm#10

The drawback is that you need GTK+ and GtkAda.

> - Server side web framework which does not require the latest GNAT to work (=works with older GNAT releases and other compilers also)

I have a HTTP server implementation. It is based on GNAT.Sockets though, and does not have any tools (I needed none for embeddable/disk-less servers where I used it).

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

If there were interest I could add an adasockets back-end. Provided adasockets support socket select.

Simon Wright probably has a HTTP server: http://embed-web-srvr.sourceforge.net

> - At least semi-decent runtime/peripheral library for ARM Cortex-Mx processors[3]

Oh, yes. As well as a cross compiler. It is a torture to use the native GNAT.

> - Interface to sqlite databases (using my own partial bindings atm.)

I have GM GPL SQLite (no C library needed) here

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

> - reStructuredText to HTML formatter

I did HTML output manually.

[...]

From: Simon Clubley <clubley@eisner.decus.org>
Date: Wed, 11 Jun 2014 12:09:23 +0000
Subject: Re: a new language, designed for safety !
Newsgroups: comp.lang.ada

> [...]

The C GTK+ libraries are LGPL which is pretty much the same thing.

[...]

> Hmm, I thought GM GPL is a least offensive license possible.

Not the least, but amongst the least. The point about bindings using a more restrictive license than the underlying library is well taken. The example which comes to mind is GTK+; the underlying C library uses the LGPL license. The ACT Ada bindings now use the GPL license.

From: Björn Lundin <b.f.lundin@gmail.com>
Date: Wed, 11 Jun 2014 05:11:23 -0700
Subject: Re: a new language, designed for safety !
Newsgroups: comp.lang.ada

> [...] Some sort of easy interface to execute programs on *nix/Windows and capture the input/output/stderr/exit code[2]

> [...]

> [2] *nix part is easy, but I have no idea how to do that on Windows :)

In GNAT for Windows, there's a file “adaint.c” in which there are functions:

```
static int  win32_wait (int *status)

static void win32_no_block_spawn
(char *command, char *args[], HANDLE
*h, int *pid)
```

which perhaps are helpful in spawning an getting exitcodes, if combined.

Ada Conformity Assessment Test Suite

From: Ada Conformity Assessment Authority
Date: Thu Jun 12 2014
Subject: Ada Conformity Assessment Test Suite (ACATS)
URL: http://www.ada-auth.org/acats.html

[...]

ACATS 4.0 is currently under development. ACATS 4.0 development snapshots are provided to give the Ada community an early look at the tests intended for inclusion in ACATS 4.0. They are formatted like an ACATS Modification List as a list of changes to be made to the existing ACATS 3.1, and an associated set of test files. This includes withdrawn (deleted) tests, modified (corrected) tests, and new tests. [...]

Interval Arithmetic

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 12 Jun 2014 22:08:32 +0200
Subject: ANN: Interval arithmetic for Ada v1.11 released
Newsgroups: comp.lang.ada

The library provides implementation of interval floating-point and integer arithmetic.

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

[See also “Interval arithmetic v1.10”, AUJ 33-1, p. 8. —sparre]

Industrial Control Widget Library

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 14 Jun 2014 11:23:40 +0200
Subject: ANN: Ada industrial control widget library v3.8 released
Newsgroups: comp.lang.ada

The library is intended for designing high-quality industrial control widgets for Ada applications. The widgets are composed of transparent layers drawn by cairo. The widgets are fully scalable graphics. A time controlled refresh policy is supported for real-time and heavy-duty applications. The library supports caching graphical operations and stream I/O for serialization and deserialization. Ready-to-use gauge and meter widgets are provided as samples as well as an editor widget for WYSIWYG design of complex dashboards. The software is based on GtkAda and cairoada, the Ada bindings to GTK+ and cairo.

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

Changes to the version 2.14:

- The library was adapted to the GtkAda 3.x. Earlier versions are no more supported;
- Only Ada 2005 and Ada 2012 are supported when GtkAda 3.x is used;
- This version was switched to the native GtkAda's cairo bindings;
- The type Interfaces.C.Double was replaced by GDouble as this is the type used in GtkAda's cairo;
- The second parameter of the procedure Refresh of the package Gtk.Layered was replaced with Cairo_Context;
- Functions Get_Time_Axis_Annotation, Get_Values_Axis_Annotation, Get_Values_Text_Angle, Get_Values_Text_Color, Get_Values_Text_Face, Get_Values_Text_Height, Get_Values_Text_Stretch, Set_Values_Text_Font were added to the package Gtk.Oscilloscope;
- Annotation text interface and labels changed to support markup;
- Compiled with GNAT 4.9.

[See also “Industrial Control Widget Library”, AUJ 33-3, p. 145. —sparre]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 28 Jul 2014 18:47:42 +0200
Subject: ANN: Ada industrial control widget library v3.9 released
Newsgroups: comp.lang.ada

aicwl is an Ada library that is intended for designing high-quality industrial control widgets for Ada applications. The widgets are composed of transparent layers drawn by cairo. The widgets are fully scalable graphics. A time controlled refresh policy is supported for real-time and heavy-duty applications. The library supports caching graphical operations and stream I/O for serialization and deserialization. Ready-to-use gauge and meter widgets are provided as samples as well as an editor widget for WYSIWYG design of complex dashboards. The software is based on GtkAda and cairoada, the Ada bindings to GTK+ and cairo.

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

Changes to the previous version:

- Oscilloscope behavior on selection can be altered using Set_Selection_Mode;
- Minor bug fixes;
- Fedora packages rely on the official GtkAda3 packages (released by Björn Persson).

Fuzzy Sets

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 18 Jun 2014 17:54:44 +0200
Subject: ANN: Fuzzy sets for Ada v5.7 released
Newsgroups: comp.lang.ada

Fuzzy sets for Ada is a library providing implementations of confidence factors with the operations not, and, or, xor, +, and *, 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 with conventional arithmetical operations, and linguistic variables and sets of linguistic variables with operations on them. String-oriented I/O is supported. A rich set of GTK+ GUI widgets is provided.

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

Changes to the previous version:

- The widgets and renderers are adapted to the GTK 3.x.
- Gtk_Fuzzy_Linguistic_Set_Domain has new style property: line-color;
- Compiled with GNAT 4.9.

[See also “Fuzzy Sets”, AUJ 33-2, p. 78. —sparre]

Zip-Ada

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Sat, 28 Jun 2014 11:51:46 -0700
Subject: Ann: Zip-Ada v.47
Newsgroups: comp.lang.ada

A new version of Zip-Ada is available [1].

The only, but, big news is that the LZMA "method" is available for decompression.

More details: [2]

[1] http://unzip-ada.sf.net/

[2] http://gautiersblog.blogspot.ch/2014/06/zip-ada-v47-with-lzma-decompression.html

[See also “Excel Writer, GNAVI, Mathpaqs and Zip-Ada”, AUJ 34-4, p. 200. —sparre]

SparForte

From: Ken Burtch <koburtch@gmail.com>
Date: Fri, 4 Jul 2014 03:40:44 -0700
Subject: ANN: SparForte 1.5
Newsgroups: comp.lang.ada

SparForte is my Ada-based shell / web template / scripting language.

This is SparForte's 14th year.

Release highlights:

- full exceptions

- teamwork features

- linked lists and hash tables packages

- web templates repaired and enhanced

- preliminary OpenGL support

SparForte can be downloaded from:

http://www.sparforte.com

A summary of the new version can be found here:

http://www.pegasoft.ca/coder/coder_june_2014.html

The change log found here:

http://www.sparforte.com/news/2014/news_sf15.html

If you enjoy SparForte, please email me and I will add your company and testimonial to the website.

[See also "SparForte", AUJ 34-4, p. 206. —sparre]

GNAT GPL for Bare Board ARM

From: AdaCore Press Center
Date: Thu Jul 24 2014
Subject: AdaCore Releases GNAT GPL for Bare Board ARM
URL: http://www.adacore.com/press/gnat-gpl-for-bare-board-arm/

Freely available toolsuite brings power of Ada language to global ARM ecosystem

NEW YORK and PARIS, (July 24, 2014)

AdaCore today released a freely downloadable version of its GNAT GPL Ada cross-development environment for Bare Board ARM Cortex processors. Students, professors and other developers of non-proprietary software can now exploit Ada 2012's reliability, safety and security benefits for ARM applications.

GNAT GPL for Bare Board ARM Cortex processors provides a complete Ada 2012 development environment, including a comprehensive tool-chain and GPS, AdaCore's flagship Integrated Development Environment (IDE).

It also includes a fully configurable/customizable run-time library consisting of the "Small Footprint" (SFP) and Ravenscar profiles that are particularly relevant to safety-critical systems.

The SFP profile corresponds to a language subset with minimal GNAT run-time routines, and the Ravenscar profile is a subset of the Ada concurrency features with an efficient, predictable, small-footprint implementation. The resulting Ada subset has expressive power well beyond that of other languages used for ARM-based devices.

"There are now billions of ARM processors in embedded systems, which has created a global ecosystem with many developers looking to take advantage of Ada's strengths," said Dr. Pat Rogers, AdaCore Bare Board product manager. "By making an Ada cross-development environment freely available to the academic and hobbyist communities, we are responding to this demand and see great potential for significantly increasing the overall usage of the Ada language. With powerful ARM-based boards currently available for under $20, this new GNAT GPL release becomes a cost-effective development environment for everyone."

The release of GNAT GPL for Bare Board ARM is part of AdaCore's ongoing commitment to the Ada community. Fully featured releases of the GNAT technology are already available for GNU Linux, Mac OS X, and Windows.

"Finally, the substantial software engineering benefits of the Ada 2012 language are available for the huge ARM microcontroller family," said Mike Silva, Software Engineer at the www.embeddedrelated.com community. "This is a ground-breaking achievement for the embedded programming world, offering the promise of higher quality embedded software delivered on shorter schedules."

"For our students, this is almost a game-changing new option, providing an academia-affordable, hands-on, high-integrity and fully real-world hardware/software environment for every individual student," said Dr. rer. nat. Uwe R. Zimmer, Fellow at the Australian National University. "Tools which enable the combination of high-integrity, real-time systems with concrete, real-world hardware will open doors to dependable, physical systems for many more students."

Availability

GNAT GPL for Bare Board ARM is available now from libre.adacore.com. The package includes a tutorial and example project showing how to use Ada and GPS for the "STM32F4 Discovery" (Cortex-M4) evaluation kit from STMicroelectronics. Additional Ada tutorials can be accessed via AdaCore University.

Qt5Ada

From: Leonid Dulman <leonid.dulman@gmail.com>
Date: Sat, 26 Jul 2014 08:22:39 -0700
Subject: Announce : Qt5Ada version 5.3.1 (372 packages) and VTKAda version 6.1.0 (656 packages) release 26/07/2014 free edition
Newsgroups: comp.lang.ada

Qt5Ada is Ada-2012 port to Qt5 framework (based on Qt 5.3.1 final Qt5ada version 5.3.1 open source and qt5c.dll (libqt5c.so) built with Microsoft Visual Studio 2012 in Windows and gcc x86-64 in Linux.

Package tested with the GNAT-GPL-2012 Ada compiler in Windows 32- and 64-bit and Linux x86-64 Debian 7.3.

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

Added Geo Navigation support (GPS & GLONASS), new packages and demos.

Qt5Ada for Windows and Linux (Unix) is available from: http://users1.jabry.com/adastudio/index.html

My configuration script to build Qt 5.3 is: configure -opensource -release -nomake tests -opengl desktop -qt-zlib -qt-libpng -qt-libjpeg -openssl-linked OPENSSL_LIBS="-lssleay32 -llibeay32" -plugin-sql-mysql -plugin-sql-odbc -plugin-sql-oci -icu -prefix "e:/Qt/5.3"

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

I hope Qt5Ada and VTKAda will be useful for students, engineers, scientists and enthusiasts.

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

[See also "Qt5Ada", AUJ 35-1, p. 7. —sparre]

Ada Web Application

From: Stephane Carrez <Stephane.Carrez@gmail.com>
Date: Sun, 27 Jul 2014 12:06:07 -0700
Subject: ANN: Ada Web Application 1.0.0
Newsgroups: comp.lang.ada

I don't post very often so I'll make a multi-announce of the following Ada libraries and components:

- Ada Utility Library: Version 1.7.1

  Download: http://download.vacs.fr/ada-util/ada-util-1.7.1.tar.gz

- Ada EL: Version 1.5.1

  Download: http://download.vacs.fr/ada-el/ada-el-1.5.1.tar.gz

- Ada Security: Version 1.1.1

  Download: http://download.vacs.fr/ada-security/ada-security-1.1.1.tar.gz

- Ada Server Faces: Version 1.0.1

  Download: http://download.vacs.fr/ada-asf/ada-asf-1.0.1.tar.gz

- Ada Database Objects: Version 1.0.1

  Download: http://download.vacs.fr/ada-ado/ada-ado-1.0.1.tar.gz

- Dynamo: Version 0.7.1

  Download: http://download.vacs.fr/dynamo/dynamo-0.7.1.tar.gz

Ada Web Application is a framework to build web applications on top of the above components as well as AWS.

The new version of AWA provides:

- New countries plugin to provide country/region/city data models
- New settings plugin to control application user settings
- New tags plugin to easily add tags in applications
- New <awa:tagList> and <awa:tagCloud> components for tag display
- Add tags to the question and blog plugins
- Add comments to the blog post

AWA can be downloaded at http://blog.vacs.fr/vacs/download.html

A small tutorial explains how you can easily setup a project, design the UML model, and use the features provided by the Ada Web Application framework.

See https://code.google.com/p/ada-awa/wiki/Tutorial

A live demonstration of various features provided by AWA is available at http://demo.vacs.fr/atlas

[See also “Ada Web Application”, AUJ 34-1, p. 11. —sparre]

POSIX Ada API

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 26 Jul 2014 21:32:59 -0500
Subject: Re: Semantics of POSIX Ada Binding
Newsgroups: comp.lang.ada

Natasha Porté wrote:

> I have been using Florist implementation of POSIX bindings for a while, and been mostly happy with it. However the documentation of Florist is a bit... terse. As far as I can tell, it amounts to "see IEEE STD 1003.5x".

> However, while both Ada and POSIX standards are freely available, it seems that versions of "IEEE STD 1003.5" are not. Or at least I have failed to find any.

Right. So far as I know, there isn't one. [...] (Not surprisingly, Janus/Ada doesn't support a POSIX binding.)

> Is there a documentation somewhere that I missed? Or are we left only with guesswork from public references?

Guesswork is about it, I fear. Unless you want to buy a copy from IEEE.

[...] don't use POSIX is the best solution. Since Ada.Directories was added to Ada, there's a lot less need to use POSIX (particularly if your implementation provides a useful Ada.Directories.Information).

> [...]

You're not alone, the problem has no solution other than to forget about using POSIX bindings.

Visual Ada Developer

From: Leonid Dulman <leonid.dulman@gmail.com>
Date: Wed, 6 Aug 2014 22:35:30 -0700
Subject: Announce: Visual Ada Developer (VAD) 7.7
Newsgroups: comp.lang.ada

VAD 7.7 Common description.

1. VAD (Visual Ada Developer) is a Tcl/Tk oriented Ada-2012 (TCL) GUI builder portable to difference platforms, such as Windows NT/Vista/7, Unix (Linux), eComStation(Os/2) and Mac OS X.

You may use it as IDE for any Ada (C, C++, TCL) project.

VAD generated Ada sources, you may compile and build executable or generate TCL script to interpretate with Tcl/Tk VAD 7.7 was tested in Windows 32bit/64bit and Linux x86-64 Debian 7.3

2. Used software

GNAT GPL 2013 Ada-12 compiler (or and others 95, 2005 or 2012)

TCL/TK 8.5.x
http://tcl.activestate.com/software/tcltk/

TCL/TK 8.6.x
http://tcl.activestate.com/software/tcltk/

You may choice needed version in link time. (I recommend to work with 8.6)

- TASH 8.02 by Terry J. Westley http://tash.calspan.com/
- IMG 1.3 package by Jan Nijtmans <Jan.Nijtmans@wxs.nl>.
- Icons 1.2 by Adrian Davis (adrian@satisoft.com)
- Help System (Html browser from Editors and Parsers menu) by Andrei A. Gratchev <grand@midc.miem.edu.ru>
- TkPaint - a simple Image Editor http://www.netanya.ac.il/~samy/tkpaint.html with pdf and eps document formats support
- RAPID-1 By Martin Carlisle
- McListbox, mcombobox by Bryan Oakley <oakley@channelpoint.com> http://purl.oclc.org/net/oakley/tcl/mclistbox/index.html
- Toplevelmanager(window::or) by Mark G. Saye
- Tktable by Jeff Hobs <jeff.hobbs@acm.org> http://www.hobbs.wservice.com/tcl/capp/
- FTP_library by Stefen Traeger <Steffen.Traeger@t-online.de> http://home.t-online.de/home/Steffen.Traeger
- Csh1.0 package by Mohamed Baccar http: //members.aol.com/~mbaccar/pub/csh10.zip
- Xterm button initialize xterminal
- Snack 2.4 multimedia sound by Kare Sjolander http://www.speech.kth.se/snack/
- BLT 3.0 ftp://ftp.tcltk.com/pub/blt
- Itcl 4.0 http://www.sensus.org/tcl/index.htm
- Tix 8.5 Tix Tcl/Tk extension
- QuickTimeTcl 3.0 multimedia movie (Quick Time for Windows and Mac) by Mats Bengtsson and Bruce O'Neel
- Tclgtk Gtk widget collection on Tcl http://tcl-gtk.sourceforge.net.
- Oratcl 4.4 Oracle connection (Oracle 9i,Oracle 10i support) http://oratcl.sourceforge.net
- Optcl 3.0 - conversion between Tcl objects and COM types by Farzad Pezeshkpour (Windows only)
- OpenGL support packages:
  - Tkogl OpenGL extension by Claudio Esperanca http://aquarius.lcg.ufrj.br/~esperanc/tkogl.html
  - tcl3d Tcl/Tk OpenGL Wrapper
  - VTK 5.x,6.x OpenGL extension by Ken Martin, Will Schroeder, Bill Lorensen http://public.kitware.com/VTK/files
  - vtkGUI 0.1 by Silvano Imboden http://visit.cineca.it/vtkGUI
  - Tkhtml 3.0 package by Dan Kennedy (last alpha 16 version has problems in tcl/tk 8.6b2. Install it only after test hv3.tcl script works !!!).
- Fve Free text editor by Kazuo Sasagawa
- Hex A simple Hex Editor by George Peter Staplin
- August Free HTML editor by Johan Bengtsson
- ASED Tcl/Tk IDE by Andreas Sievers
- Dom, TclXML XML parser http://www.zveno.com
- Whiteboard 0.94.3 Image and Media Viewer by Mats Behgtsson

- TkMC File manager by Grigoriy Abramov
- IDL_To_Ada_Translator by Scott R. Bennet http://www.mitre.org
- TCL/TK XML intelligence Visual Editor by Alexander V.Dederer http://tkxmlive.sourceforge.net
- TCL/TK InstallJammer Multiplatform Installer http://www.installjammer.com/

VAD is available from http://users1.jabry.com/adastudio/index.html

GNAT GPL for Various STM32F4xx Boards

From: Jerry Petrey <gpetrey@earthlink.net>
Date: Sat, 16 Aug 2014 13:05:58 -0700
Subject: Re: GNAT SPARK:Embedded ARM Ada Project doesn't run in STM32F429 Discovery Board
Newsgroups: comp.lang.ada

> [...]

1. I have it running on the STM407 Discovery board and on the STM417 Discovery Board but have not been able to get it on the STM429 board yet - that chip is a bit different.
2. You have to look in the lib->gnat->arm-eabi->ravenscar-sfp-stm32f4 directory for the startup files. In adainclude, the file setup_pll.adb will need some changes as well as s-stm32f.ads, a-intnam.ads, handler.S, s-bbbosu.adb, and s-bbpara.ads as far as I can tell. Also the linker script stm32f4-rom.ld in the ada adalib directory will need a change due to the larger Flash memory.
3. The clocks are setup in setup_pll.adb
4. I think I am getting close to having it run on the 429 board but I am not there yet. On the other boards it is great. I have console I/O working, as well as USARTs, interrupts and DMA working.

Ada and Operating Systems

Mac OS X: GNAT

From: Simon Wright <simon@pushface.org>
Date: Sat, 24 May 2014 18:00:14 +0100
Subject: ANN: GCC 4.9.0 (2014) for Max OS X Mavericks
Newsgroups: comp.lang.ada

GCC 4.9.0, with the GNAT GPL 2014 tools, is available at

https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/4.9.0-2014/

This is the README:

This is GCC 4.9.0 built for Mac OS X Mavericks (10.9.2, Darwin 13.1.0), with Xcode 5.1.1 and tools from GNAT GPL 2014.

gcc-4.9.0-x86_64-apple-darwin13-2014.tar.bz2

Compilers included: Ada, C, C++, Objective C, Objective C++, Fortran.

Tools included: ASIS, AUnit, GDB, GNATColl, GPRbuild, and XMLAda from GNAT GPL 2014.

Target: x86_64-apple-darwin13

Configured with: ../gcc-4.9.0/configure \

```
--prefix=/opt/gcc-4.9.0 \
--disable-multilib \
--disable-nls \
--enable-languages=c,c++,ada,fortran,
  objc,obj-c++ \
--host=x86_64-apple-darwin13 \
--target=x86_64-apple-darwin13 \
--build=x86_64-apple-darwin13
```

Thread model: posix

gcc version 4.9.0 (GCC)

MD5 (gcc-4.9.0-x86_64-apple-darwin13-2014.tar.bz2) = 4f8e94f0349757ecd417e97b604ce99e

Install by

$ cd /

$ sudo tar jxvf ~/Downloads/gcc-4.9.0-x86_64-apple-darwin13-2014.tar.bz2

and put /opt/gcc-4.9.0/bin first on your PATH.

Installing GDB

gdb has to be 'code-signed' (unless you're willing to run it as root!)

Instructions are at

https://gcc.gnu.org/onlinedocs/gnat_ugn_unw/Codesigning-the-Debugger.html

Notes

The compiler is GPL version 3 with the Runtime Exception, so executables built with it can be released on proprietary terms PROVIDED THAT they make no use of the packages from GNAT GPL 2014, which are full GPL.

The command 'gnat', as originally built, failed with SIGSEGV. It was rebuilt on its own, using the project file gnatcmd.gpr, and no longer failed; the working version is provided.

Changes made to GPRbuild GPL 2014 are in gprbuild-gpl-2014-src.diff. They:

- remove the '-c' flag that is wrongly passed to ranlib (and isn't by gnatmake).
- correct a problem when building static stand-alone libraries.
- remove some restrictions not provided in FSF GCC yet: No_Fixed_IO, No_Long_Long_Integers, No_Multiple_Elaboration.
- import the new library package GNAT.Rewrite_Data (used by gprslave).
- retain the bug in gprinstall which installs executables with 'execute' access for the owner only rather than for all users (this is a problem if the installation is done by root). The change relies on a change in the RTS (adaint.c).
- gprslave can't call Set_File_Last_Modify_Time_Stamp (adaint.c again).

GNATColl GPL 2014 was configured as below, which is minimal apart from GNU Readline being enabled. Users may wish to reconfigure for their own requirements.

```
./configure \
  --prefix=/opt/gcc-4.9.0 \
  --build=x86_64-apple-darwin13 \
  --enable-gpl
```

resulting in

```
Shared libraries: yes (default: static)
Gtk+: no
(requires pkg-config and gtkada.gpr)
Python: yes
/System/Library/Frameworks/
Python.framework/Versions/2.7
(see --with-python)
PyGtk: no (see --enable-pygtk)
PyGObject: no (see --enable-pygobject)
Syslog: yes (see --enable-syslog)
Readline (GPL license): yes
(see --with-readline --enable-gpl)
Gmp: no (see --with-gmp)
PostgreSQL: no -L/usr/lib
(see --with-postgresql)
Sqlite: embedded (see --with-sqlite)
Iconv: yes (see --with-iconv)
Projects: yes
```

Changes to ASIS GPL 2014 are in asis-gpl-2014-src.diff. Only changes necessary for the build are included.

GDB GPL 2014 built without changes, but there are problems with 'catch exception'; one workround is to invoke GDB with the '-readnow' switch. See https://sourceware.org/bugzilla/show_bug.cgi?id=11385

In addition to the above, a new library gnat_util is required by ASIS and GNATColl. A Sourceforge project to provide this has been set up at https://sourceforge.net/projects/gnatutil/; release 4.9.0 is included here. This is the equivalent of the Debian libgnatvsn.

From: Simon Wright <simon@pushface.org>
Date: Sat, 31 May 2014 20:35:16 +0100
Subject: Re: ANN: GCC 4.9.0 (2014) for Max OS X Mavericks

Newsgroups: comp.lang.ada

> [...]

The first release (gcc-4.9.0-x86_64-apple-darwin13-2014.tar.bz2) contained an error: 'gnat list' didn't recognise library projects. This meant that Emacs ada-mode couldn't be used for library projects.

gcc-4.9.0-x86_64-apple-darwin13-2014-1.tar.bz2 is now available at the same place as above.

Debian and Fedora: GtkAda

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 1 Jun 2014 22:01:23 +0200
Subject: GtkAda 3.8.2 binary packages
Newsgroups: comp.lang.ada

[...] Debian (deb) and Fedora (rpm) packages can be downloaded here:

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

Mac OS X: XAdaLib

From: Pascal <p.p14@orange.fr>
Date: Tue, 3 Jun 2014 20:40:27 +0200
Subject: [ANN] XAdaLib 2014 binaries for MacOS 10.9 including GTKAda 3.8 and more.
Newsgroups: gmane.comp.gnome.gtk+.ada

This is XAdaLib 2014 built on MacOS X 10.9 Mavericks for X11 including:

- GTK Ada 3.8.2 with GTK+ 3.8.4 complete,
- Glade 3.10.2,
- GnatColl GPL 2014,
- Florist GPL 2014,
- AdaCurses 20110404,
- Gate 3-04-b

to be installed for instance at /usr/local:

$ cd /usr/local

$ sudo tar xzf xadalib-gpl-2014-x11-x86_64-apple-darwin13.2.0-bin.tgz

Update your PATH to include gtkada-config, glade, gate3.sh and other executables in it:

$ PATH=/usr/local/xadalib-2014/bin:$PATH

Update your GPR_PROJECT_PATH to include gtkada.gpr, adacurses.gpr, florist.gpr, gnatcoll.gpr and other projects in it:

$ export GPR_PROJECT_PATH=/usr/local/xadalib-2014/lib/gnat:$GPR_PROJECT_PATH

Then see documentation and examples in share directory and enjoy.

Coming soon, the instructions which have produced the libraries on Blady web site:

http://blady.pagesperso-orange.fr/creations.html#gtkada

And the Ada industrial control widget library from Dmitry Kazakov.

XAdaLib binaries have been post on Source Forge:

http://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2014-mavericks/

Feel free to send comments on the list.

From: Pascal <p.p14@orange.fr>
Date: Mon, 9 Jun 2014 10:50:33 +0200
Subject: Re: [ANN] XAdaLib 2014 binaries for MacOS 10.9 including GTKAda 3.8 and more.
Newsgroups: gmane.comp.gnome.gtk+.ada

Here are the instructions I used to build GTKAda on MacOS (in French):

http://blady.pagesperso-orange.fr/telechargements/gtkada/Install-GTKAda-X11.pdf

Here are the modifications I made:

http://blady.pagesperso-orange.fr/telechargements/gtkada/xadalib-2014-diff.tgz

I haven't built icu4c, is it needed by gtkada?

Debian: Transition to GNAT 4.9

From: Emilio Pozuelo Monfort <pochu@debian.org>
Date: Sun, 10 Aug 2014 16:54:58 +0200
Subject: Re: Bug#756078: transition: gnat
To: 756078@bugs.debian.org, debian-ada@lists.debian.org, Reto Buerki <reet@codelabs.ch>

> [...]

The last blockers are #756081 and #755076. Can someone from debian-ada take a

look at those?

From: Nicolas Boulenguez <nicolas.boulenguez@free.fr>
Date: Mon, 18 Aug 2014 02:40:19 +0200
Subject: status update
To: Ada in Debian mailing list <debian-ada@lists.debian.org>

[...]

Libgnatcoll is waiting for manual approval in the NEW queue. This may take a while because it is a new source package. Asis and libaws are ready and will follow quickly.

It is now possible to work on adabrowse, adacontrol or libalog. Here are the steps to build the required libgnatcoll/asis/aws snapshots.

You may either create an empty Monotone database

mtn -d $db db init

or use an existing one.

Download the packaging from the server to your database. The first time you contact this server, monotone will show its key. You should check that it matches f8a11727e8983cf9083c08c6a2acaa27e439dd39.

mtn -d $db pull mtn://www.ada-france.org?org.debian.libgnatcoll

Last steps depend on the package.

mtn -d $db cat debian/README.source | less

Microsoft Windows: Dokan

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 16 Aug 2014 18:12:12 +0200
Subject: ANN: Dokan Ada bindings v 1.0
Newsgroups: comp.lang.ada

Dokan library for developing is user-space file system for Windows. The Ada bindings are Ada 95 compliant. A sample implementation of a memory-resident file system is included.

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

References to Publications

HOLWG Submissions?

From: J. Kimball <jkimball4@gmail.com>
Date: Fri, 21 Mar 2014 11:04:25 -0500
Subject: HOLWG submissions
Newsgroups: comp.lang.ada

Are there any digital copies of the Yellow or Blue manuals that were submitted to DoD?

From: Nasser M. Abbasi <nma@12000.org>
Date: Sun, 23 Mar 2014 03:14:48 -0500
Subject: Re: HOLWG submissions
Newsgroups: comp.lang.ada

> [...]

"Request for proposals were issued April 1977; 17 proposals received. Four contractors were picked to produce prototype languages:

- Cii Honeywell Bull led by Jean Ichbiah (green)
- Intermetrics led by Benjamin M. Brosgol (red)
- SofTech led by John Goodenough (blue)
- SRI International led by Jay Spitzen (yellow)"

So the names of the persons are above. May be this can help in googling them more?

Here is info on the red one:

http://henrylivingston.com/maida/computer/redref/index.htm

Carl Brandon Selected as Top Innovator

From: Embedded Computing Design
Date: Tue Jun 10 2014
Subject: Embedded Computing Design selects Top Innovators, Most Influential Women for June issue
URL: http://embedded-computing.com/21408-embedded-computing-design-selects-top-innovators-most-influential-women-for-june-issue/

[...]

Dr. Carl Brandon, Professor, Vermont Technical College

Dr. Brandon worked for IBM for two years, designing their first memory chip with two colleagues. He has been teaching at Vermont Technical College since 1977, and while there has introduced Pascal and Ada programming courses, and helped set up the Computer Tech and Aeronautical Engineering Technology degrees. Starting in 2004, he started the CubeSat Lab, resulting in the launch of the first university CubeSat from a college in New England.

[...]

[Carl and his students used Ada and SPARK to program their CubeSat. -- sparre]

Programming in Ada 2012

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri Jun 20 2014
Subject: "Programming in Ada 2012" is now available
URL: http://www.adaic.org/2014/06/programming-in-ada-2012/

"Programming in Ada 2012" by John Barnes is now available. This is the first book that we're aware of that specifically covers Ada 2012. "Programming in Ada 2012" is an update of John's previous book, "Programming in Ada 2005", which was an update of "Programming in Ada 95", which was an update of "Programming in Ada". What we can tell from this (besides the fact that John has been involved in Ada since the beginning) is that this is a useful and widely used book (of ever expanding size!) that has withstood the test of time. We expect that this new edition will be the same.

The book can be ordered at Amazon [1] and presumably other booksellers. It also can be ordered directly from the publisher [2].

[1] http://www.amazon.co.uk/Programming-Ada-2012-John Barnes/dp/110742481X

[2] http://www.cambridge.org/dk/academic/subjects/computer-science/software-engineering-and-development/programming-ada-2012

alt.embedded on GNAT-GPL for ARM Cortex-M4

From: William Wong <bill.wong@penton.com>
Date: Mon Jul 7 2014
Subject: Running Ada 2012 On The Cortex-M4
URL: http://electronicdesign.com/blog/running-ada-2012-cortex-m4

I like to think I write good code, and I've used C and C++ almost since their inception. I admit to incorporating more than one unwanted bug into C applications that were eliminated after sometimes tedious diagnostic sessions. Almost every new microcontroller released has a free C/C++ compiler toolchain associated with it.

Unfortunately, C is very unforgiving, and C++ is only a little better. But they are the mainstay for embedded programmers these days. That's one reason why I have been waiting for AdaCore's delivery of its Ada 2012 toolchain for Arm's Cortex-M platform. It is a free download at libre.adacore.com.

[...]

Blog Entries on STM32F4 Programming

From: Mike Silva <embeddedrelatedmike@scriptoriumdesigns.com>
Date: Tue, 5 Aug 2014 19:29:06 -0700
Subject: Finally, Ada on $15 hardware
Newsgroups: comp.lang.ada

As some may know, and others not, AdaCore has released a libre version of its ARM Cortex M3/M4 port (maybe other models too, I didn't look). Out of the box it runs on an STM32F4 board (the $15 hardware referred to above).

I've written a couple of tutorial blog entries on this:

http://www.embeddedrelated.com/showarticle/617.php

I for one am quite delighted to finally have Ada on a excellent (and cheap!) 32-bit microcontroller family.

Ada Inside

Hospital Information System

From: AdaCore Press Center
Date: Wed May 28 2014
Subject: SmartWard Pty Ltd Selects AdaCore Tools for Hospital Information System Development
URL: http://www.adacore.com/press/smartward-hospital-information-system/

Ada chosen for benefits in reliability, safety, and security

MELBOURNE, Australia, NEW YORK and PARIS – May 28, 2014 – Australian System Safety Conference – AdaCore today announced the adoption of its GNAT Pro Ada Development Environment and CodePeer static analysis tool by the Australian healthcare informatics company SmartWard Pty Ltd for use in implementing its state-of-the-art patient care management system. The SmartWard system needs to be highly reliable and secure from unauthorized access, it has to provide real-time response and 24x7 availability, and it also must be easy to use by hospital staff. After evaluating alternative potential approaches, the company selected the Ada language and AdaCore software development tools as the best solution for meeting these requirements.

The SmartWard system replaces a paper-based, manual approach that is time-consuming and error prone. It runs on computers at each patient bedside and at all other points-of-care, providing up-to-date information on scheduled activities, patient alerts and vital signs and allowing real time entry of treatment records. It presents patient histories in user-friendly charts with decision support data, and validates medication and patient identity automatically via smart sensors.

With its long history of successful usage for many types of safety-critical and high-security software, Ada was chosen as the implementation language for the SmartWard system. Many errors that would only be detected through significant debugging effort in other languages are caught at compile time in Ada, and features such as Ada 2012's contract-based programming help embed low-level requirements into the source program as assertions that can be checked at run time or verified statically.

AdaCore's GNAT Pro development environment, along with several complementary tools, is being used to implement the SmartWard software. With its sophisticated data- and control-flow analysis, the CodePeer automated code review and validation tool helps in identifying potential logic errors, including "off by 1" bugs in loops and other more subtle problems. CodePeer's static analysis can be conducted both during a system's initial development, and also retrospectively to find potential vulnerabilities in existing code. Another AdaCore tool that is proving useful to SmartWard is the Ada Web Server (AWS). Its web-socket implementation is being used for communication between the SmartWard system's front-end and back-end.

"Different language technologies have different strengths," said Cyrille Comar, AdaCore Managing Director. "Ada was specifically designed for systems where the concept of a 'fatal error' may be

literally true, and we're pleased to see Ada adopted for medical applications such as SmartWard where reliability, safety and security are so critical."

"The use of Ada has helped us significantly in instilling a safety culture within our company," said Dr. Malte Stien, CTO of SmartWard. "We see Ada as a competitive advantage in our market, and the use of the language is a selling point for our product."

[...]

About SmartWard Pty Ltd

SmartWard is an innovative health informatics company founded in 2009. It has worked closely with nurses and hospitals since then to develop a unique new system that delivers much-needed improvements in the efficiency of hospitals and aged care facilities, while improving quality-of-care. SmartWard is now commercializing this system.

A clinical trial completed in 2013 has proven the SmartWard proposition. It showed that SmartWard allowed the nursing staff to double the amount of time they were able to spend with their patients, by completely replacing the paper-based system with digitized records and by moving the record access/update site from the nurses' workstation to the patient's bedside. SmartWard also reduced the time for the shift handover while improving the accuracy of the provided care.

Wireless Temperature Sensor

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Fri, 27 Jun 2014 09:14:53 +0300
Subject: Usage of AVR-Ada (Was: Lcd and arduino nano)
To: avr-ada-devel@lists.sourceforge.net

[...] I have now had AVR-Ada based wireless temperature sensor running on my balcony almost one month (the device is Olimexino-328 with custom XBee shield, powered by single 1000mAh lipo).

I plan to write about it, but I am still waiting for the battery to run out - not sure how many weeks I need to wait. :)

HTTP File Server

From: Natasha Porté <lithiumcat@instinctive.eu>
Date: Mon, 7 Jul 2014 20:43:27 +0000
Subject: ANN: HTTP file server v1.0-beta1
Newsgroups: comp.lang.ada

I had stopped posting announcement about my personal projects here because it seemed useless and made me feel like a lunatic rambling in the desert about things completely disconnected from reality. But now I'm trying very hard to convince myself that there's a small possibility that somebody here might find it useful.

So this is a program based on AWS, meant to solve the problem of transferring a file from one computer to another, when the most convenient way to do so is using an intermediary HTTP server (e.g. when both computer are deep behind NAT routers and/or stringent firewalls). That fact it happens so often is a testament to the sad state of IT nowadays.

And it obviously requires the ability to run a custom HTTP server, which unfortunately restricts severely the potential audience.

More specifically, it is designed for the following scenarios:

1. the service owner wants to make a file available to one or several people, sharing a download link,

2. anybody wants to send a file to the service owner while avoiding abuse, modelled as the following scenario:

3. somebody makes a file available to somebody else without knowledge or consent from the service owner.

To achieve this, the uploader can only access a report page, to ensure the file has been correctly uploaded. To compute the download link, a server-wide secret is required, presumably only known by the service owner.

More details are available on the full-length project description page, on the official fossil repository page at [1] and on the GitHub mirror at [2].

This version is labelled "beta" because I have been running it in production for a while now, without finding any fault, but so far to the best of my knowledge nobody else has tried it, and the fact that by myself I haven't found any bug in my own code does not mean much.

So if you're interested in the application, please let me know, I will gladly consider any bug report or feature request.

If not, would please at least have a look at the documentation pages I linked above, and tell me whether it's clear and makes sense and allows someone to deploy it, or whether there are some parts obscure or missing.

I would also gladly read any comments or reviews about my code or the concept, but that's much more than I dare hoping.

[1] http://fossil.instinctive.fr/simple-webapps/doc/tip/README.md

[2] https://github.com/faelys/simple-webapps

$6 \cdot 10^{12}$ m on 170k Lines of Ada 83

From: Airbus Defence & Space
Date: Wed Aug 6 2014
Subject: Rosetta comet chaser reaches destination to decipher origins of solar system 4,6 billion years ago
URL: http://airbusdefenceandspace.com/rosetta-comet-chaser-reaches-destination-to-decipher-origins-of-solar-system-46-billion-years-ago/

Rosetta comet chaser reaches destination to decipher origins of solar system 4,6 billion years ago

- After a journey of six billion kilometres, satellite and lander embark on the search for primary matter
- Mission confirms Airbus Defence and Space's role as a key ESA partner in planetary research

The Rosetta comet chaser developed and built by Airbus Defence and Space for the European Space Agency (ESA) has arrived at its destination after flying for more than 10 years and travelling more than six billion kilometres. It is now ready to swivel into an orbit around the comet called 67P/Churyumov-Gerasimenko. The mission assigned to Rosetta and to the Philae lander that it is carrying, is to examine primary material from the nursery of the solar system 4.6 billion years ago over the next one and a half years.

At 11.30 CET, the ESA satellite control centre in Darmstadt, Germany, received news via radio signal that Rosetta had begun its approach. The space probe's rendezvous with 67P took place some 400 million kilometres away from Earth. Rosetta will now accompany the comet on its journey around the sun and back again into the depths of our solar system. The Philae lander is scheduled to touch down on the comet's surface in November in order to carry out measurements there.

[...]

From: Andreas Jung and Jean-Loup Terraillon
Date: Wed Dec 8 2010
Subject: Faster, Later, Softer: COrDeT – an on-board software reference architecture
URL: http://flightsoftware.jhuapl.edu/files/2010/FSW10_Jung.pdf

[...]

Software size of the central computer's ESA missions is increasing...

- Science satellites
  - Exosat (launch 1983), RCA1802 – 8K memory, ASM
  - SOHO (launch 1995), 2xMDC281 – 2x64KB , Ada83
  - Rosetta (launch 2004), 2xMA3-1750 – 2x1MB, 170KLoc, Ada83

[...]

Ada in Context

Ada 202X Wish List

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 26 Mar 2014 15:41:30 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...] the [[wide_]wide_]string packages and the relationship to [[wide_]wide_]character -- it would be very nice (as well as aiding maintainability) to have the *_String [and character-handling] packages be generics instantiated on the proper Character type.

I agree that this is an area that needs looking at, but I don't think using more generics will provide anything useable. The problem, as Dmitry likes to say, is that the representation and semantics of a string are intertwined, and those have to be separated in order to make a sensible string type system.

I've played with some ideas based on an abstract Root_String'Class, and pretty much everything necessary can be done with existing Ada 2012 features, and the few things that can't have a fairly obvious language feature that could be defined to provide them (for instance, a mechanism to support string literals).

I think the problem is mainly going to be political rather than technical. The solution requires defining a large set of new packages that echo functionality already in the language, and that would not be used by the sorts of safety-critical applications that the paying customers use. (They're not using Text_IO or Unbounded_Strings or Directories or ...). That's going to make changes in this area a tough sell, I fear. Hope I'm wrong.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 27 Mar 2014 16:50:20 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...] Care to share your findings?

Of course really. The rough sketch is in the !discussion of AI12-0021-1. Please note that we've never discussed this AI within the ARG, so I might be the only one interested in pursuing this idea. I've fleshed it out further mentally; in particular, one could imagine supporting conversions through a common type (in this case Wide_Wide_String).

The primary "problem" is that in this model, most strings become tagged and communicate using Wide_Wide_Character and Wide_Wide_String. That probably seems more inefficient than it actually is (after all, Unbounded_Strings are already tagged -- controlled, actually -- and this couldn't be less efficient than that, so long as the language-defined types stick to single inheritance). For the characters, the representation doesn't matter much (and on a 32-bit machine, they're all the same performance anyway). So the main issue is the cost of converting to-from Wide_Wide_String. Perhaps there is some other way to ensure universal convertability which doesn't require performance-sapping multiple inheritance or multidispatching.

After all, if storage space is not a concern, Wide_Wide_String is the most universal (and efficient) representation. It seems that it makes the most sense to do operations in terms of that type and convert for storage -- since that's what will happen naturally most of the time. (Presuming you're not so American-centric that you don't care about anything beyond type String. :-)

Anyway, lots more thought needed.

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Thu, 27 Mar 2014 18:54:39 -0700
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...]

I guess storage space is not a concern for Erlang. The string "ABC" is shorthand for the list [65, 66, 67], and apparently each "character" takes 8 bytes: 4 for the number, and 4 for the pointer to the next element in the list

Using 4 bytes for the character is sometimes called support for Unicode.

Why not define Character as 32 bits and get rid of the Wide_ guys?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 28 Mar 2014 09:17:42 +0100
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...] most strings become tagged and communicate using Wide_Wide_Character and Wide_Wide_String.

In another model only Wide_Wide_String'Class would be tagged.

Real issue: Classes of non-tagged types.

> So the main issue is the cost of converting to-from Wide_Wide_String.

You would not need conversions if the specific operations were provided rather than inherited (as they are provided presently).

Real issue: Multi-methods (String vs String) and full multiple dispatch (String vs Character)

To reiterate the point. The implementation of strings in Ada is all OK, it is the interface to this implementation which sucks.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 28 Mar 2014 16:27:09 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> Real issue: Multi-methods (String vs String) and full multiple dispatch (String vs Character)

That's an alternative, but the question is whether that can be implemented with less overhead than the scheme I suggested. I believe the answer is no, at least within generalized string packages (which hopefully will become the norm for new string operations in Ada). It's surely one of the questions to be considered - nothing I think on this topic (or any topic, for that matter) is likely to be the last word.

> [...] implementation of strings in Ada is all OK, it is the interface to this implementation which sucks.

Right, but that manifests itself in duplicated and overly restrictive packages, which would have to be reworked in order to use a more general interfaces. (Ada.Strings.Bounded and Ada.Strings.Unbounded in particular have overloaded operations that would make everything ambiguous if not eliminated.)

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 31 Mar 2014 18:55:07 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

[...]

"Root_String_Type" would be abstract, and the other types would be derived from it. New types corresponding to each interesting representation would be defined. The existing types would still exist but be obsolescent.

[...]

> Whether Unbounded_String should become a member of this new hierarchy or be replaced there with a new type is another question.

It's not a question. There's no way to do that because the operations already defined for Unbounded_String would make it ambiguous if given string literals (and all Root_String_Type'Class types would have string literals). It would be completely unusable.

As such, all new types is the only possibility. That's what makes this idea politically messy.

[...]

The problem is that Unbounded_String defines operations like

```
function "&" (Left : Unbounded_String;
              Right : String)
   return Unbounded_String;
```

which would be ambiguous with the normal "&" if both String and Unbounded_String had string literals (as they must for Root_String_Type to work). Indeed, that's the only reason that Ada doesn't have a way to define string literals for a private type -- we talked about it years ago but determined that it cannot be used with any of the existing string packages. As such, it would have been a weird thing to define.

We can't get rid of these problematical operations -- it would be way too incompatible. So new packages is the only way to go.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 26 Mar 2014 16:06:18 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...]

Commenting here is precisely the problem. Most of the user contacts that the ARG has is third-hand at best. (Combating that is one of the reasons I hang out here.) We really don't have the clear of an idea of what other Ada users want. I know I draw on my own experience far more than probably is healthy.

Moreover, we've been asking for years for users to show us the problems that they cannot solve with Ada (now Ada 2012), so that we can look at possible solutions. That's one of the major purposes of the Ada-Comment mailing list. A few people do that, but it's not many. You don't have to be a language designer to do that; indeed, we'd rather you didn't try to invent language features to solve the problem, since those ideas often obscure the actual problem. Of course, we might determine that Ada 2012 already can solve the problem sensibly - or that we don't consider it important enough - but problems that we don't hear about have a near 100% chance of not being addressed.

This is probably a good time to remind everyone that the Ada-Comment mailing list is open to everyone. It's for comments and discussion on the Ada language and it's Standard, not particular implementations of Ada. (Leave your gripes about AdaCore here, please. :-) You can find instructions about joining the list at:

http://www.adaic.org/resources/add_content/standards/articles/comment.html

Of course, that is not to say that discussion here isn't useful when one wants to know if they are alone with a concern or the like. But please don't assume that just because something is said here it is going to get back to the ARG and have an impact on future language standards. (Occasionally, someone has taken a discussion here back to Ada-Comment, but no one ought to assume that will happen. If you really care, do it yourself!)

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue, 29 Apr 2014 17:26:19 +0300
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

Not sure have it been proposed anywhere, but a common ABI (application binary interface) for the most popular platforms (32-bit ARM Linux, 64-bit x86_64/amd64 Linux, 32-bit i386 Windows, 64-bit x86_64/amd64 Windows) would be nice. This way I could use binary libraries generated by one Ada compiler (version) with another Ada compiler (version) without providing full sources (or recompilation).

Another thing is the lack of "modern" general purpose libraries for net/www stuff, embedded devices, various algorithms/data structures, etc. Those libraries won't necessarily need to be part of the standard. It is enough if they are just somehow freely available without extra cost.

From: François Fabien <francois_fabien@hotmail.com>
Date: Sun, 30 Mar 2014 05:28:05 -0700
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

One wish to ease I/O: To have some equivalent of this C pattern:

```
printf("%c %c %c %c %c \n %c %c %c
       %c %c \n",
```

From: Pascal Obry <pascal@obry.net>
Date: Sun, 30 Mar 2014 21:02:00 +0200
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...]

Well, this is possible in Ada with some coding of a package, one could write something like (maybe already implemented by someone, I remember a simple_io package or something like that):

```
Formatted_Print
   ("%c %c %c %c %c \n %c
     %c %c %c %c \n" & Var1 & Var2
    & Var3 & Var4 & Var5 & Var6
    & Var7 & Var8);
```

You just have to code the Formatted_Print routine once for all. No big deal:

```
type Formatted_String is private;

Format_Error : exception;

function "&" (F : Formatted_String;
              V : Character)
      return Formatted_String;

function "&" (F : Formatted_String;
              V : Integer)
      return Formatted_String;
function "&" (F : Formatted_String;
              V : Float)
      return Formatted_String;
...

procedure Formatted_Print
      (F : Formatted_String);

procedure Formatted_Print
      (File : Text_IO.File_Type;
       F : Formatted_String);
```

From: Britt <britt.snodgrass@gmail.com>
Date: Wed, 2 Apr 2014 09:21:20 -0700
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

[...]

Require basic support for 64-bit integer types:

Standard.Long_Long_Integer (I'm currently missing this in ObjectAda)

Interfaces.Integer_64

Interfaces.Unsigned_64

and corresponding support for C99's "long long", "int64_t" and "uint64_t" in Interfaces.C.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 2 Apr 2014 17:53:25 -0500
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...]

Even on 8-bit and 16-bit processors?

(Note that no one should be depending on implementation-defined Standard types like Long_Long_Integer. These are considered implementation-defined by the profile No_Implementation_Extensions; they should be avoided in portable Ada code. Also see 3.5.4(28). I have more sympathy for requiring System.Max_Int to be at least 2**63-1 on appropriate machines (and the appropriate Interfaces types as well)) -- but what's an appropriate machine? Janus/Ada has never supported 64-bit integers, mainly because of the nasty effect on shared generics for formal integer and discrete types [it would force those to use all 64-bit math, which would get pretty expensive for Text_IO.Integer_IO and similar generics]. We only support 32-bit machines, but of course modern Intel processors all have 64-bit "integers" via the floating point instruction set. I've never considered that real 64-bit support and ignored it, because of expense and precision concerns [along with complications supporting indexing and for loops] -- not worth the headache. A 64-bit target would be a different beast, of course.)

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Wed, 02 Apr 2014 17:01:15 -0700
Subject: Re: Your wish list for Ada 202X
Newsgroups: comp.lang.ada

> [...]

I'd like the language to support any integer type declaration a user is willing to write, regardless of "efficiency". For some such declarations, the compiler can chain together multiple machine "words" as we used to do in the Good Old Days® to get 16-bit integers on 8-bit machines like the 6502. At some the point the compiler should be allowed to use the same representation as Unbounded_Integer, which would implement the unlimited-precision integer package the compiler writer has had to use to write the compiler (or a similar package for the target for cross compilers).

Not going to happen, I know, but ...

I'd also like a mode in which objects that don't fit on the stack would be automatically put on the heap.

Portable Bindings to Opaque C Types

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue May 6 2014
Subject: Easy way to create portable bindings to opaque C types
URL: http://ada.tips/easy-way-to-create-portable-bindings-to-opaque-c-types.html

Have you ever struggled to create bindings for C types like FILE or CURL? If yes, then you know how hard it can be, since the type might not be same on all platforms.

An easy solution for these is to define a null record which represents the type and use only access types (pointers) when handling the type.

For example:

```
type CURL_Type is null record;
type CURL_Access is access all
        CURL_Type;

function curl_easy_init return
        CURL_Access;
pragma Import (C, curl_easy_init,
        "curl_easy_init");

CURL_Obj : CURL_Access;
...
CURL_Obj := curl_easy_init;
```

This works on all platforms, all compilers, and all relatively recent Ada variants (95, 2005, 2012 at least).

Of course, if you need to change the internals of the C variables from Ada, this approach does not work.

Termination of Tasks Waiting on a Protected Queue

From: Natasha Porté <lithiumcat@instinctive.eu>
Date: Sun, 18 May 2014 07:32:17 +0000
Subject: Termination of tasks waiting on a protected queue
Newsgroups: comp.lang.ada

I have been having a task termination issue, and I'm wondering whether I got my design wrong or whether I'm missing something, so I ask you to help me on that.

The basic need is delegating potentially long jobs to a dedicated task (or pool of tasks) so that the program flow generating the jobs and continue quickly. I furthermore assume that the jobs are "fire and forget", that there is no need to report anything about completion (or lack of thereof) to the code that generated the jobs (or more realistically, that reporting is performed through channels outside of the scope of the problem).

So I went with the basic design below:

```
package Example is
  type Job_Description is private;

  function Create_Job (...)
        return Job_Description;
  procedure Enqueue_Job
        (Job : in Job_Description);

private

  package Job_Lists is new
       Ada.Containers.Doubly_Linked_Lists
        (Job_Description);

  protected Queue is
    procedure Append
        (Job : in Job_Description);
    entry Get_Next
        (Job : out Job_Description);
  private
    List : Job_Lists.List;
    Has_Job : Boolean := False;
  end Queue;

  task Worker is
  end Worker;
end Example;

package body Example is
  procedure Enqueue_Job
        (Job : in Job_Description) is
  begin
    Queue.Append (Job);
  end Enqueue_Job;

  protected body Queue is
    procedure Append
        (Job : in Job_Description) is
    begin
      List.Append (Job);
      Has_Job := True;
    end Append;

    entry Get_Next
        (Job : out Job_Description)
      when Has_Job is
    begin
      Job := List.First_Element;
      List.Delete_First;
      Has_Job := not List.Is_Empty;
    end Get_Next;
  end Queue;

  task body Worker is
    Job : Job_Description;
  begin
    loop
      Queue.Get_Next (Job);
      -- <actually to the job>
    end loop;
  end Worker;
end Example;
```

As you might have guessed, I have recently read a lot of material about Ravenscar profile, and as far as I can tell this example does match the profile, even though the original need happens in a much more relaxed environment.

For example, without Ravenscar restrictions, the Worker task could easily be turned into a task type, and use an array of them to implement a pool of workers.

The problem is, how to terminate cleanly the worker tasks in a non-Ravenscar environment when the main application is completed?

But then, how can a worker task entry be used to solve my problem? A protected operation cannot call a task entry, because it's potentially blocking. The job generator cannot call the entry directly, because it would block when the task is not ready, so I still need a queue between the generator and the worker task.

Alternatively, I could use a special value for Job_Description (or a new out parameter) to make the worker exit its infinite loop, and somehow make the protect Queue object aware that the master is completed, so that it can signal the worker task(s) to complete. But how can this be implemented? Since the tasks block the finalization of the master, I can't rely on a Finalize procedure to notify the protected object.

From: Brad Moore <brad.moore@shaw.ca>
Date: Sun, 18 May 2014 17:05:59 -0600
Subject: Re: Termination of tasks waiting on a protected queue
Newsgroups: comp.lang.ada

> [...] so I still need a queue between the generator and the worker task.

A protected entry can however requeue to a task entry.

I was faced with a similar problem in the non-Ravenscar task pools in Paraffin.

I did not want the programmer to have to call some protected subprogram to trigger the task pool to terminate. I also did not want to have to wait for a timeout to expire before the application could exit. I wanted it to be immediate.

So in your example, it might look something like;

```
package body Example is
   function Create_Job
         return Job_Description is
      Result : Job_Description;
   begin
      return Result;
   end Create_Job;

   procedure Enqueue_Job
         (Job : in Job_Description) is
   begin
      Queue.Append (Job);
   end Enqueue_Job;

   protected body Queue is
      entry Append
         (Job : in Job_Description) is
      begin
         if not Idle then
            requeue Worker.Work_Queued;
         else
            List.Append (Job);
         end if;
      end Append;

      procedure Get_Next
          (Job : out Job_Description) is
      begin
         Job := List.First_Element;
         List.Delete_First;
      end Get_Next;

      function Is_Empty return Boolean is
      begin
         return List.Is_Empty;
      end Is_Empty;

      procedure Worker_Idle is
      begin
         Idle := True;
      end Worker_Idle;

      function Worker_Is_Idle
         return Boolean is
      begin
         return Idle;
      end Worker_Is_Idle;
   end Queue;

   task body Worker is
      Job : Job_Description;
   begin
      loop
         begin

            Queue.Worker_Idle;

            if not Queue.Is_Empty then
               Queue.Get_Next (Job);
            else
               select
                  accept Work_Queued
                    (Next_Job : Job_Description)
                  do
                     Job := Next_Job;
                  end Work_Queued;
               or
                  terminate;
               end select;
            end if;

            --  <actually to the job>
         end;
      end loop;
   end Worker;
end Example;
```

Representation Units?

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Wed, 21 May 2014 00:22:08 +0200
Subject: spec/body/rep (Was: Compilation error (GNAT bug?))
Newsgroups: comp.lang.ada

[...]

What if the "external aspects" went elsewhere? For example, in a representation unit. (That's a name I remember). With representation units, the source text proper becomes more portable, and more configurable at the same time. There is no need to change aspects in the source text when switching environments or when changing the configuration (Linker_Options is one example). Just pick a suitable representation unit. Declarations also become more readable, insofar as they'd focus on just the logic, not link names and such.

Representation units can provide an Ada version of dependency injection, even when injection happens at compile time.

In fact, GNAT already supports "outsourcing" certain aspects with the help of project files. Do some of the other compilers do that, too?

Since aspects are a fairly new addition to the language, chances are that representation units will not generate backwards compatibility issues.

Would representation units help producing clear separation of concerns?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 23 May 2014 16:21:21 -0500
Subject: Re: spec/body/rep (Was: Compilation error (GNAT bug?))
Newsgroups: comp.lang.ada

[...]

Huh? A "representation unit" or whatever you call it is part of the "source text". You can't run the program without it.

That's the problem with fancy project management (no matter how well-designed) -- it's part of the program (you can't build it without it), but it's outside of the language definition. (And it would be impractical to add it to the language definition.) So it causes vendor lock-in.

> [...] Would representation units help producing clear separation of concerns?

I don't think so, mainly because you already have such a capability: constants! It's not the presence or absence of an aspect that changes, it's the value. So it might make sense to have a package specifically for the target-specific details (many systems do that, including the ACATS).

For instance:

```
package Target_Specific is
   -- Package for Windows.
   pragma Linker_Options (...);

   -- Data types:
   type Largest_Integer is
      range -2**31 .. 2**31 with Size => 32;
   type Largest_Modular is
      mod 2**32 with Size => 32;

   -- Interfacing details for package Blarch:
   Foo_External_Name :
         constant String := "...";
   Bar_External_Name :
         constant String := "...";
   ...
end Target_Specific;
```

And then have separate versions of the package for the various targets supported. I don't see any benefit to creating a new kind of unit (with the massive costs that would have for compilation systems) just to reproduce capabilities that already exist.

(For what's it's worth, I don't believe that it makes sense to separate representation from other aspects (pun intended) of a declaration. All of these things have fundamental impacts on the semantics of an entity, and trying to deny that (as the Ada 83 designers attempted to) just leads to a forest of odd restrictions and complex rules designed to keep a fiction going while still allowing a simple compiler design. [The majority of the freezing rules come about because of this desire, for instance.] And it isn't even a very useful fiction. See type Largest_Integer above; if we need to give that a different size on some other target, we need to change the range, too. That's pretty common when dealing with representation.[Disclaimer: My personal views here may not be held by others, even within the ARG.])

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 27 May 2014 07:16:56 +0200
Subject: Re: spec/body/rep (Was: Compilation error (GNAT bug?))
Newsgroups: comp.lang.ada

> [...]

> package Target_Specific is

> [...]

And to ease porting, have a package called Target_Specific_Windows, another one called Target_Specific_Linux. All the users do:

```
with Target_Specific;
```

and have the following library-level renaming:

```
with Target_Specific_Windows;
package Target_Specific renames
      Target_Specific_Windows;
```

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Tue, 27 May 2014 09:22:16 +0300
Subject: Re: spec/body/rep (Was: Compilation error (GNAT bug?))
Newsgroups: comp.lang.ada

> [...]

I don't see what advantage such as a library-level renaming gives. If one is developing for several platforms, say Windows and Linux, there will then be two library-level renamings somewhere, one as above and the other using Target_Specific_Linux, but some compiler-specific way is still needed to choose which of the library level renamings to include in the compilation. So one could just as well call both the target-specific packages Target_Specific, directly, and use the same compiler-specific way to choose which one to compile. For example, I use GNAT's ADA_INCLUDE_PATH to choose the folder ("linux" or "windows") which contains the version of Target_Specific to be compiled.

Would there be some sense in being able to specify such library-level renamings as configuration pragmas? This might give us a standard way to choose component versions depending on the configuration (leaving as compiler-specific the way to select which configuration is to be compiled... :-)

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 27 May 2014 10:54:21 +0200
Subject: Re: spec/body/rep (Was: Compilation error (GNAT bug?))
Newsgroups: comp.lang.ada

> [...]

The way I do it, both renamings are in the same file, one of them commented out. I just comment/uncomment the right one at the time of build. Not fully automated, but easy, and I argue (with the C people) that it is hardly more work than changing a global variable in a Makefile.

The point is: one single simple change in one file, and your whole application switches OSes. The other benefit being that you see quite well which parameters are for which OS.

But of course, it all depends on your build process, use case, and personal taste...

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 27 May 2014 10:55:39 +0200
Subject: Re: spec/body/rep (Was: Compilation error (GNAT bug?))
Newsgroups: comp.lang.ada

> [...]

I have issues with this approach in general. The problem is that there is not any check that the interfaces of the target-specific packages are same in the sense that all target-independent clients are compilable with any "implementation."

In my projects I, of course, use neither pragmas nor renaming. It is too clumsy and unmaintainable. I simply put same named package into different target-specific directories, e.g. x86/windows or i686/linux and switch them using gpr scenario.

This does not solve the abovementioned problem, though. As a possible solution, without introducing some huge stuff of formal package interfaces, it would be enough to be able to switch only the private part of the specification and the package body, keeping the public part same. It would not work with target-specific constants and conditionally with-ed packages.

[...] target-specific packages could be considered instances of some virtual generic package with target as an actual parameter.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Tue, 27 May 2014 17:45:07 +0200
Subject: Re: spec/body/rep (Was: Compilation error (GNAT bug?))
Newsgroups: comp.lang.ada

[...]

A configuration pragma seems to have the property that just one may cover many units, whereas library level renamings lack this formal connection, and there may be many. (Directories, or discipline, providing for a more or less formal mode of development.)

When the compiler knows about "representation units" (I think Bob Duff once mentioned such a thing using this name), and the language ties them to (the private part of) a unit, then at least programmers will have something explicit and reliable, issues notwithstanding:

> [...] The problem is that there is not any check that [...] all target-independent clients are compilable with any "implementation."

Assuming that a universal expression of "compilability" in any configuration is nice, but likely impossible, [...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 27 May 2014 17:57:38 -0500
Subject: Re: spec/body/rep (Was: Compilation error (GNAT bug?))
Newsgroups: comp.lang.ada

"G.B." <rm-dash-bauhaus@dash.futureapps.de> wrote in message

news:5384b302$0$6663$9b4e6d93@newsspool3.arcor-online.net...

...

> When the compiler knows about "representation units" [...]

But this solves nothing. There has to be some implementation-defined (or project-defined) way of selecting which "representation unit" is selected for a particular compilation. And that's the problem, with any of these solutions. (I agree with Dmitry about the problem of keeping the versions of the packages in sync. I believe this has to be solved by the version-control; one of the reasons that I find typical VCs useless is that they refuse to solve that problem and solve other unlikely problems instead.)

In any case, adding a new kind of unit would require sweeping changes to the language standard and to implementations. It would require a pretty significant problem to even consider such a change. We did in fact consider that for the mutually-dependent package problem, but ultimately decided to avoid it in favor of the "virtual" limited view solution. If we're unwilling to use such a solution to solve a critical problem, I can hardly imagine using it to solve a problem that's not hard to solve with some tools (as with Dmitry's project manager solution) or discipline (as in J-P's comment in or out of a library-level renames).

Style: Discriminants or Components?

From: Edward R. Fish <onewingedshark@gmail.com>
Date: Mon, 02 Jun 2014 20:44:24 -0600
Subject: Style Question: normal record vs discriminated null-record.
Newsgroups: comp.lang.ada

Ok, given some sort of alert-system (meaning there shouldn't be any/much in the way of value manipulation of those objects) is it preferable to use a discriminated null-record or a record with normal components?

```
type Grievousness is (Warning, Error);
type Circumstance is (Expired_Data,
        Bad_Request, Malformed_Data);

type Alert_1 (Severity  : Grievousness;
              Condition :  Circumstance)
        is null record;

type Alert_2 is
  record
    Severity  : Grievousness;
    Condition : Circumstance;
  end record;
```

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 03 Jun 2014 06:38:37 +0200
Subject: Re: Style Question: normal record vs discriminated null-record.
Newsgroups: comp.lang.ada

> [...]

Depends how you perceive your data. Discriminants are intended to define some kind of subclasses of your type, and can be used to parameterized inner

components. Some criteria for you to chose:

- You can define subtypes according to certain values of discriminants.
- Discriminants are always initialized.
- Discriminants cannot be changed after the declaration.
- You can have a private type whose discriminants are visible (and components private)

From: Edward R. Fish <onewingedshark@gmail.com>
Date: Tue, 03 Jun 2014 00:20:04 -0600
Subject: Re: Style Question: normal record vs discriminated null-record.
Newsgroups: comp.lang.ada

> Depends how you perceive your data. [...]

Right -- the particular situation [alerts] seems like a good fit because of these qualities:

(1) Subtypes for critical failures can be defines and quick/clear disambiguation/determination in handling [an unconstrained type] via in'.

(2) Which means, ideally that the point that generates the alert always generates correctly.

(3) That they cannot be altered means that to 'change' the message is to handle it and generate a new one -- kind of like "renaming"/re-raising exceptions.

This is what makes it seem like a good fit, but I was still wondering what "normal Ada programmers" thought about it.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 3 Jun 2014 09:19:42 +0200
Subject: Re: Style Question: normal record vs discriminated null-record.
Newsgroups: comp.lang.ada

> [...]

My rule of thumb - discriminants whenever possible, components as a fallback.

I wish Ada allowed discriminants of any type.

Identifying Solutions in Recursive Solvers

From: Mike Hopkins <postmaster@ada-augusta.demon.co.uk>
Date: Thu, 5 Jun 2014 18:49:01 +0100
Subject: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

Purely for my own amusement I have written a Sudoku puzzle solver. The program is written In Ada. However being in my ninth decade I am firmly stuck in an Ada95 time-warp. As a general rule, Sudoku puzzles rated as 'gentle' or 'moderate' can be solved by systematic elimination of alternatives. But these relatively simple deterministic methods can be expected to fail when confronted by a puzzle rated as tough' or a 'diabolical'. When this happens in my program, trial and error is invoked. It is rare that more than four or five trial and error passes are required.

The enclosing subprogram is called recursively by the trial and error process. Each recursive call adds a pair of nodes onto an implicit but invisible binary tree (the run-time call stack). What I was hoping was that on detected that the solution has been found, it would be possible to return to, and exit from, the main program by simply exiting each recursive call in turn in order climb back up the recursion ladder. However, each step of that climb is a re-entry into the caller where there may be remaining unfinished business. The solution having been found, that unfinished business is now known to be no more than garbage. Nevertheless an elegant solution might be expected to clear garbage before that caller re-enters its own caller.

The only solution that I can see is to jump straight out of the tree. But that seems to lack elegance. The jump is made by raising an exception which has been declared in, and is handled by, the enclosing subprogram. The exception is 'silent' because the handler contains a null statement.

I fear that perhaps I am missing something but have no idea what.

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 5 Jun 2014 11:30:29 -0700
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> [...]

Without seeing an actual program or any code at all, I can't really tell, but ... when a caller calls itself recursively, isn't there either a function result or an OUT parameter that allows the callee to tell the caller whether it has succeeded? In which case the caller simply exits, and returns to *its* caller passing back the correct answer and if necessary a flag indicating that it's succeeded. I have no idea whether I've identified the problem correctly, but it's the best I can do without seeing any code. Anyway, I think that's the general approach to handling backtracking problems.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 05 Jun 2014 21:00:28 +0200
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> [...] The jump is made by raising an exception which has been declared in, and is handled by, the enclosing subprogram. The exception is 'silent' because the handler contains a null statement. [...]

[...] I think exceptions are perfectly appropriate for that: they allow to unwind the call stack directly up to the point where you want to catch it by providing a handler.

I know that not everybody likes this idea, but to me exceptions are a powerful programming structure, not limited to handling errors.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 05 Jun 2014 21:43:07 +0200
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> As the name indicates, exceptions are for exceptional situations. Finding a solution doesn't seem exceptional for a solver.

It IS an exceptional condition!

An exceptional condition is a condition that makes it impossible or unnecessary to continue with the normal algorithm, and requires suddenly a different behaviour. Finding the solution is precisely that. (although it is not an error or an abnormal condition, which is precisely the point I made).

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Thu, 05 Jun 2014 19:12:55 -0400
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> [...]

I'm with J-P here. There are cases where exceptions can reasonably be used for not-so-exceptional cases. Maybe the OP's Sudoku solver is one such -- I haven't seen the code, so I don't know.

> I know that not everybody likes this idea, ...

That's somewhat of an understatement; many people are quite passionate about it, and say things like "Never use exceptions for control flow".

But exceptions ARE control flow. When an exception is raised, a transfer of control to the handler happens (or to the end of the program if unhandled). It is impossible to use exceptions other than for control flow!

Others add the word "normal": "Don't use exceptions for normal control flow". Unfortunately, it's unclear what "normal" means.

> ...but to me exceptions are a powerful programming structure, not limited to handling errors.

I agree. IMHO, the purpose of exceptions is to deal with the case where one piece of code detects an error (or maybe just an unusual situation), and a different piece of code knows what to do about it (or even to decide it's not an error after all).

"end of file" might be considered an error by the file-reading procedure, but might be considered perfectly normal by the

caller. So I don't think it makes sense to say "exceptions are ONLY for errors" -- different pieces of code have different views on whether it's an error.

In any case, if you need to jump out of many layers of (recursive?) calls, an exception might well be the best way. Checking error codes at each level might be verbose and error prone.

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 5 Jun 2014 16:39:47 -0700
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> In any case, if you need to jump out of many layers of (recursive?) calls, an exception might well be the best way. Checking error codes at each level might be verbose and error prone.

I don't like it. But if you do something like this, I'd suggest that this use be limited to an exception that you declare inside a subprogram, so that you raise and handle it only inside that subprogram or nested subprograms. Otherwise, someone could look at a subprogram that is called in between, and never guess that the subprogram might not complete normally (A calls B, B calls C, C raises an exception that gets passed over B's head back to A; a programmer trying to read B might not suspect that B may not complete in a non-error situation.) In other words, keep such usages as localized as possible.

Another thing to keep in mind is that exceptions cause overhead. I've seen implementations that have to do some stuff any time a subprogram or a block with an exception handler is entered. I've seen other implementations that, in order to eliminate this overhead in "normal" (non-exception) cases, perform table lookups on each address in the stack until it finds a handler; this is a relatively expensive operation that those implementations have decided is justified because exceptions aren't supposed to happen in "normal" cases. Whether this overhead is less than the expense of going through a number of returns, I don't know--I'm sure it depends on various factors. But efficiency should not be a reason to use exceptions instead of straight returns, because it may well make things slower.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Sat, 07 Jun 2014 08:03:23 +0200
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> [...]

The whole thing boils down to the difference between "normal" and "exceptional". FWIW, here is how I explain it in my courses:

A program is basically looping (if it were to do things just once, it would be faster by hand than writing a program). The loop is the general case, the "rule". Sometimes, you encounter a condition that cannot be handled by the normal "rule" and requires a different treatment: it is an "exception" to the rule. That's why it's called exception, and not trap, abnormality, failure...

From: Brad Moore <brad.moore@shaw.ca>
Date: Fri, 06 Jun 2014 08:13:30 -0600
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> [...] efficiency should not be a reason to use exceptions instead of straight returns, [...]

Another point to keep in mind is that although the exception mechanism may or may not be technically faster than the normal recursion exit, depending on implementation, it may not be noticeably faster. A guideline I try to follow generally is to write the code naturally and simply, and let the compiler worry about performance, and then only look at using different constructs if there is still a performance problem that needs to be addressed.

I have actually written a Sudoku solver, that executes in Parallel. My approach was to just let the recursion unwind naturally.

In my parallelism framework, workers tasks catch and handle exceptions, where if exceptions are raised in multiple worker threads, only one of those exceptions is saved, and gets reraised in the calling thread before returning from the parallel call.

Here exceptions are generally used to report failures, but could probably be used to report a solution in this case. However if a different exception occurs in a worker thread (such as a constraint error), that may or may not be the exception that ends up getting reported.

I suspect that trying to use exceptions to report solutions for Sudoku, would not noticeably improve performance, as most of the time is spent trying to find a solution, not report it.

My Sudoku solver uses a brute force approach. (I was mostly interested in trying out the parallelism). I believe the performance could be improved significantly by updating local cells to maintain a list of possible values, thus ruling out trial values much more sooner. I would think such an approach would improve performance far more than using exceptions to exit recursion, so that would be where I would suggest programming effort be spent.

I have several versions of the solver:

- A sequential version,

- A load balancing version,

- A load balancing version that adds some stack safety (prevents stack overflow)

[Source text for sequential Sudoku solver not included here. --sparre]

From: Mike Hopkins <postmaster@ada-augusta.demon.co.uk>
Date: Thu, 5 Jun 2014 21:03:10 +0100
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> [...]

Thank you for giving me an alternative angle of view.

The trial and error process is optimised by preselection of a matching pair of cells, they match in the sharing of a common pair of candidate solutions. By analogy, it is a choice of left or right. On average, 50% of trials will have explored both possibilities in which the first will have been wrong and the second will have proved to be correct. In the other 50%, the first choice will have been correct so the second choice is left dangling and unvisited..

I chose, perhaps wrongly, that the complete grid of 81 cells is passed down the recursion tree (as IN OUT). At each level, the grid is further completed as the full gamut of deterministic algorithms is exercised before either a lack of further success prompts a further trial and error attempt or a positive failure forces a return of up the tree. Thus, a return of control to a caller is taken as a positive signal of failure. But, currently, there is no equivalent positive indicator of success.

I am beginning to think that a three-state flag is required (NO, MAYBE, YES). YES is not known until the 81st cell is solved. The job is now complete and control must now be passed back up the tree. What is different is that a YES would be a tangible way of contradicting the previous assumption that a return of control means failure.

From: Adam Beneschan <adam@irvine.com>
Date: Thu, 5 Jun 2014 13:40:57 -0700
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> [...] a return of control to a caller is taken as a positive signal of failure. [...]

My own experience with problems of this sort has been that passing the entire problem-state to the next level as an IN OUT parameter (or the equivalent, in other languages) makes life more difficult. The callee needs to be able to make changes to the problem-state in order to try different possibilities; the caller needs the problem-state to stay the same, so that if the callee returns without finding a solution, the caller can try the next thing on the problem-state it was given as an input parameter. The last time I wrote a Sudoku solver, I therefore made sure the grid I passed to recursive calls

was a (modified) copy of the input parameter, not a reference to the same parameter. For some kinds of problems, this might be unfeasible; but an array of 81 integers is pretty easy to handle. By the way, the program found answers almost instantaneously, and it used only backtracking--no attempt to try to deal with simple cases the way I would approach it if solving by hand. So while it might be interesting (and instructive) to try to write a program that will work optimally, in practice it isn't necessary for this particular game.

Good luck. Backtracking programs aren't easy to write correctly.

From: Stefan Lucks <stefan.lucks@uni-weimar.de>
Date: Fri, 6 Jun 2014 11:10:43 +0200
Subject: Re: OT: A bit of Sudoku
Newsgroups: comp.lang.ada

> My own experience with problems of this sort has been that passing the entire problem-state to the next level as an IN OUT parameter (or the equivalent, in other languages) makes life more difficult.

It really depends on the problem you are trying to solve.

> The callee needs to be able to make changes to the problem-state in order to try different possibilities; the caller needs the problem-state to stay the same, so that if the callee returns without finding a solution, the caller can try the next thing on the problem-state it was given as an input parameter.

Right. So the callee *must* undo the change(s) it made, before returning without a solution. If undoing is that trivial, then an in-out parameter (or "global" variable declared in some outer scope) for the state is reasonable. It may even be useful to transfer a solution found back to the callee (just don't undo the changes you made ...).

As I understand for the Sudoku case, the entire change is to assign a digit to an empty cell, and undoing means to turn the cell's state back to empty. If I am right, undoing changes is very easy, indeed!

> The last time I wrote a Sudoku solver,

Well, I have never written a Sudoku solver, but I did apply the above approach to other backtracking problems, see, e.g., <http://rosettacode.org/wiki/Knight%27s_tour#Ada>.

BTW, a Sudoku-solver for Ada is still missing at Rosetta Code http://rosettacode.org/wiki/Sudoku.

Self-referential Types

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 1 Aug 2014 14:44:57 -0500
Subject: Re: We should introduce aliased types
Newsgroups: comp.lang.ada

> [...] Every value of an aliased type should be aliased [...]

You do know that this is already true of almost all limited types (without any declaration)?

I.e.

```
type T;
type Ptr_Holder (D : access T) is
      limited null record;
type T is limited record
   Ptr : Ptr_Holder (T'Access);
end record;
```

is legal Ada.

It's not allowed for non-limited types because assigning the object would break the self-referential link (it would point to the wrong object afterwards).

And you can't directly declare a component as

```
Ptr : access T := T'Access;
```

because of accessibility; the problem isn't whether T is aliased or not. “access T” is a library-level type, while T'Access might refer to a local object, so the access type could outlive the object (in particular, this component could be copied into an object of another library-level type), and that's not allowed.

[...]

Default Values

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Sun, 03 Aug 2014 18:07:35 +0200
Subject: Default values (Was: Quick question regarding limited type return syntax)
Newsgroups: comp.lang.ada

>> LRM 3.5(56.3/3)

> It's only for scalar types though.

That's because the one for arrays is named Default_Component_Value - and record components can have their default values declared without aspects.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 4 Aug 2014 16:29:39 -0500
Subject: Re: Default values (Was: Quick question regarding limited type return syntax)
Newsgroups: comp.lang.ada

> [...]

And access types are already default initialized to null, protected type components are like record components, and there's nothing visible in a task that needs initialization.

It's not a perfect solution in that you can't change the default initialization of an access type nor of an array type with non-scalar components, but it ensures that it is possible to provide (or have one by definition) a real default initialization for every type.

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.

2014

October 01-02 — 8th **International Parallel Tools Workshop**, Stuttgart, Germany. Topics include: tools for debugging and performance tuning of parallel applications.

October 02-03 — 14th **International Conference on Quality Software** (QSIC'2014), Dallas, Texas, USA. Topics include: software testing, software quality (review, inspection and walkthrough, reliability, safety and security, ...), static and dynamic analysis, validation and verification, economics of software quality, formal methods, component software and reuse, component-based systems, cyber-physical systems, distributed systems, embedded systems, safety critical systems, etc.

October 06-09 — 33rd **International Symposium on Reliable Distributed Systems** (SRDS'2014), Nara, Japan. Topics include: distributed objects and middleware systems, experimental or analytical evaluations of dependable distributed systems, formal methods and foundations for dependable distributed computing, high-assurance and safety-critical distributed system design and evaluation, secure and trusted distributed systems, etc.

October 12-15 — 20th **International Symposium on Distributed Computing** (DISC'2014), Austin, Texas, USA. Topics include: theory, design, implementation, modeling, analysis, or application of distributed systems and networks; concurrency, synchronization; fault tolerance, reliability, availability; specification, verification, and testing: tools, methodologies; etc.

October 12-16 — 9th **International Conference on Software Engineering Advances** (ICSEA'2014), Nice, France. Topics include: advances in fundamentals for software development; advanced mechanisms for software development; advanced design tools for developing software; software security, privacy, safeness; specialized software advanced applications; open source software; agile software techniques; software deployment and maintenance; software engineering techniques, metrics, and formalisms; software economics, adoption, and education; improving productivity in research on software engineering; etc.

October 15-16 — 6th **International Workshop on Software Engineering for Resilient Systems** (SERENE'2014), Budapest, Hungary. Topics include: requirements engineering & re-engineering for resilience; frameworks, patterns and software architectures for resilience; verification, validation and evaluation of resilience; empirical studies in the domain of resilient systems; etc.

♦ Oct 18-21 — ACM SIGAda **Annual International Conference on High Integrity Language Technology** (HILT'2014), Portland, Oregon, USA. Sponsored by ACM SIGAda, in cooperation with Ada-Europe and the Ada Resource Association. Co-located with SPLASH 2014.

☺ October 20-24 — ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2014), Portland, Oregon, USA. Topics include: all aspects of software construction and delivery, at the intersection of programming, languages, and software engineering.

October 20 21st **International Workshop on Foundations of Object-Oriented Languages** (FOOL'2014). Topics include: language semantics, type systems, program verification, concurrent and distributed languages, language-based security issues, etc.

☺ October 20 5th Annual **Workshop on Evaluation and Usability of Programming Languages and Tools** (PLATEAU'2014). Topics include: methods, metrics and techniques for evaluating the usability of languages and language tools, such as empirical studies of programming languages, methodologies and philosophies behind language and tool evaluation, software design metrics and their relations to the underlying language, user studies of language features and software engineering tools, critical comparisons of programming paradigms, tools to support evaluating programming languages, etc.

☺ October 20 1st **Workshop on Software Engineering for Parallel Systems** (SEPS'2014). Topics include: parallel design patterns, modeling techniques for parallel software, parallel programming models and paradigms, refactoring and reengineering for parallelism, testing and debugging of parallel applications, tools and environments for parallel software development, case studies and experience reports, etc.

October 21-24 14th **International Conference on Formal Methods in Computer-Aided Design** (FMCAD'2014), Lausanne, Switzerland. Co-located with MEMOCODE'2014 and DIFTS'2014. Topics include: theory and application of formal methods in computer-aided design and verification of computer systems and related topics; 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; etc.

November 03-06 25th IEEE **International Symposium on Software Reliability Engineering** (ISSRE'2014), Naples, Italy. Topics include: reliability, availability, and safety of software systems; validation, verification, testing and dynamic analysis; software quality and productivity; software security; dependability, survivability, and resilience of software systems; open source software reliability engineering; supporting tools and automation; industry best practices; empirical studies; etc.

November 03-07 16th **International Conference on Formal Engineering Methods** (ICFEM'2014), Luxembourg, Luxembourg. Topics include: abstraction and refinement; program analysis; software verification; formal methods for software safety, security, reliability and dependability; tool development, integration and experiments involving verified systems; formal methods used in certifying products under international standards; formal model-based development and code generation; etc.

November 04-06 14th **International Conference on Software Process Improvement and Capability dEtermination** (SPICE'2014), Vilnius, Lithuania. Topics include: process assessment, improvement and risk determination in areas of application such as automotive systems and software, aerospace systems and software, medical device systems and software, safety-related systems and software, financial institutions and banks, small and very small enterprises, etc.

November 16-21 27th **International Conference for High Performance Computing, Networking, Storage and Analysis** (SC'2014), New Orleans, Louisiana, USA. Topics include: parallel algorithms, applications, distributed computing, performance, programming systems, system software, state-of-the-practice, etc.

November 16-22 22nd ACM SIGSOFT **International Symposium on the Foundations of Software Engineering** (FSE'2014), Hong Kong, China. Topics include: architecture and design; components, services, and middleware; distributed, parallel, and concurrent software; embedded and real-time software; formal methods; model-driven software engineering; program analysis; reverse engineering; safety-critical systems; scientific computing; software engineering education; software evolution and maintenance; software reliability and quality; specification and verification; tools and development environments; etc.

☺ Nov 16-17 3rd International **Conference on Multicore Software Engineering, Performance, and Tools** (MUSEPAT'2014). Topics include: software engineering for multicore (CPU or GPU) and heterogeneous systems; specification, modeling and design of multicore systems; programming models, languages, compiler techniques and development tools for multicore; parallel and distributed testing and debugging (PADTAD); software maintenance and evolution of multicore systems; performance tuning and optimization of multicore; domain- and platform-specific multicore software issues in scientific computing, embedded and mobile systems.

November 17-19 12th **Asian Symposium on Programming Languages and Systems** (APLAS'2014), Singapore. Topics include: foundational and practical issues in programming languages and systems, such as semantics, design of languages and type systems, domain-specific languages, compilers, interpreters, abstract machines, program analysis, verification, model-checking, software security, concurrency and parallelism, tools and environments for programming and implementation, etc.

November 27-28 **European Conference Software Engineering Education** (ECSEE'2014), Seeon Monastery, Germany. Topics include: new methods, techniques, best practices, and experiences in SE education; illustrative examples to highlight SE topics in education; tools for SE education, both commercial and public domain; etc.

December 01-04 21st **Asia-Pacific Software Engineering Conference** (APSEC'2014), Jeju Island, Korea. Topics include: embedded real-time systems; formal methods; SE environments and tools; security, reliability, and privacy; software engineering methods; software maintenance and evolution; software process and standards; testing, verification, and validation; etc.

December 08-12 15th ACM/IFIP/USENIX **International Middleware Conference** (Middleware'2014), Bordeaux, France. Topics include: design, implementation, deployment, and evaluation of distributed system platforms and architectures for computing, storage, and communication environments, including reliability and fault-tolerance; scalability and performance; programming frameworks, parallel programming, and design methodologies for middleware; methodologies and tools for middleware design, implementation, verification, and evaluation; etc.

☺ December 09-11 15th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2014), Hong Kong. Topics include: all areas of parallel and distributed computing; reliability and fault-tolerance, formal methods and programming languages, software tools and environments, parallelizing compilers, component-based and OO technology, parallel/distributed algorithms, task mapping and job scheduling, high-performance scientific computing, etc.

December 12 Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

December 10-12 15th **International Conference on Product Focused Software Development and Process Improvement** (PROFES'2014), Helsinki, Finland. Topics include: software engineering techniques, methods, and technologies for product-focused software development and process improvement as well as their practical application in an industrial setting. Deadline for submissions: November 3, 2014 (posters).

☺ December 16-19 20th IEEE **International Conference on Parallel and Distributed Systems** (ICPADS'2014), Hsinchu, Taiwan. Topics include: parallel and distributed applications and algorithms, middleware, multi-core and multithreaded architectures, scheduling, security and privacy, dependable and trustworthy computing and systems, real-time systems, cyber-physical systems, embedded systems, etc.

December 17-20 21st IEEE **International Conference on High Performance Computing** (HiPC'2014), Goa, India. Topics include: parallel and distributed algorithms/systems, parallel languages and programming environments, hybrid parallel programming with GPUs and accelerators, scheduling, resilient/fault-tolerant algorithms and systems, scientific/engineering/commercial applications, compiler technologies for high-performance computing, software support, etc. Deadline for early registration: November 14, 2014.

2015

January 04-06 14th **International Conference on Software Reuse** (ICSR'2015), Miami, Florida, USA. Topics include: domain-specific languages; COTS-based development and reuse of open source assets; software product line techniques; generative development, model-driven development; software composition and modularization; software evolution and reuse, and reengineering for reuse; quality assurance for software reuse, such as testing and verification; reuse of non-code artifacts (process, experience, etc.); transition to software reuse and industrial experience with reuse; etc.

January 08-10 16th IEEE **International Symposium on High Assurance Systems Engineering** (HASE'2015), Daytona Beach, Florida, USA. Topics include: tools and techniques used to design and construct systems that, in addition to meeting their functional objectives, are safe, secure, and reliable.

January 13-14 POPL2015 - ACM SIGPLAN **Workshop on Partial Evaluation and Program Manipulation** (PEPM'2015), Mumbai, India. Topics include: program and model manipulation techniques (such as: partial evaluation, slicing, symbolic execution, refactoring, ...); program analysis techniques that are used to drive program/model manipulation (such as: abstract interpretation, termination checking, type systems, ...); techniques that treat programs/models as data objects (including: metaprogramming, generative programming, embedded domain-specific languages, model-driven program generation and transformation, ...); etc. Application of the above techniques including case studies of program manipulation in real-world (industrial, open-source) projects and software development processes, descriptions of robust tools capable of effectively handling realistic applications, benchmarking.

January 19-21 10th **International Conference on High Performance and Embedded Architectures and Compilers** (HiPEAC'2015), Amsterdam, the Netherlands. Topics include: computer architecture, programming models, compilers and operating systems for embedded and general-purpose systems; parallel, multi-core and heterogeneous systems; reliability and real-time support in processors, compilers and run-time systems; architectural and run-time support for programming languages; programming models, frameworks and environments for exploiting parallelism; compiler techniques; etc.

March 04-06 7th **International Symposium on Engineering Secure Software and Systems** (ESSoS'2015), Milan, Italy. Topics include: automated techniques for vulnerability discovery and analysis; programming paradigms, models, and domain-specific languages for security; verification techniques for security properties; security by design; static and dynamic code analysis for security; processes for the development of secure software and systems; etc.

March 04-06 23rd Euromicro **International Conference on Parallel, Distributed and Network-Based Computing** (PDP'2015), Turku, Finland. Topics include: embedded parallel and distributed systems, multi- and many-core systems, programming languages and environments, runtime support systems, performance prediction and analysis, shared-memory and message-passing systems, dependability and survivability, real-time distributed applications, etc.

March 09-13 **Design, Automation and Test in Europe Conference** (DATE'2015), Grenoble, France. Topics include: real-time programming languages and software; formal models for real-time systems; worst case execution time analysis; tools and design methods for real-time, networked and dependable systems; dependable systems including safety and criticality; software for safety critical systems; compilers for embedded multi-core, heterogeneous, GPU, reconfigurable, or FPGA platforms; certified compilers; verification techniques for embedded systems ranging from simulation, testing, model-checking, SAT and SMT-based reasoning, compositional analysis and analytical methods; theories, languages and tools supporting model-based design flows covering software, control and physical components; modeling, design, architecture, optimization, and analysis of Cyber-Physical Systems (CPS); case studies in CPS ranging from automotive systems, and avionics, to smart buildings and smart grids; etc.

March 16-19 14th **International Conference on Modularity** (Modularity'2015), Ft. Collins, Colorado, USA. Topics include: varieties of modularity (generative programming, aspect orientation, software product lines, components, ...); programming languages (support for modular abstraction in: language design; verification, specification, and static program analysis; compilation, interpretation, and runtime support; formal languages; ...); software design and engineering (evolution, empirical studies of existing software, testing and verification, composition, methodologies, ...); tools (refactoring; evolution and reverse engineering; support for new language constructs, ...); applications (distributed and concurrent systems; middleware; cyber-physical systems; ...); complex systems; composition; etc. Deadline for submissions: October 3, 2014 (Modularity Visions track abstracts), October 10, 2014 (research papers round 2, Modularity Visions track papers).

♦ March 24-27 28th GI/ITG **International Conference on Architecture of Computing Systems** (ARCS'2015), Porto, Portugal. Focus: "reconciling parallelism and predictability in mixed-critical systems". Topics include: models and tools for multi-/many-core systems including but not limited to programming models, runtime systems, middleware, and verification; design, methods, and hardware and software architectures for mixed-critical systems; architectures and design methods/tools for robust, fault-tolerant, real-time embedded systems; etc. Deadline for submissions: October 6, 2014 (papers), November 3, 2014 (workshops, tutorials).

April 11-18 18th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2015), London, UK. Events include: CC (International Conference on Compiler Construction), ESOP (European Symposium

on Programming), FASE (Fundamental Approaches to Software Engineering), FOSSACS (Foundations of Software Science and Computation Structures), POST (Principles of Security and Trust), TACAS (Tools and Algorithms for the Construction and Analysis of Systems). Deadline for submissions: October 10, 2014 (abstracts), October 17, 2014 (full papers).

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

☺ April 13-17 **Track on Programming Languages** (PL'2015). Topics include: compiling techniques, domain-specific languages, formal semantics and syntax, garbage collection, language design and implementation, languages for modeling, model-driven development, new programming language ideas and concepts, practical experiences with programming languages, program analysis and verification, programming languages from all paradigms, etc.

☺ April 13-17 **Track on Object-Oriented Programming Languages and Systems** (OOPS'2015). Topics include: aspects and components, code generation and optimization, distribution and concurrency, formal verification, integration with other paradigms, software evolution, language design and implementation, modular and generic programming, secure and dependable software, static analysis, testing and debugging, type systems, etc.

☺ April 13-17 **Track on Software Engineering** (SE'2015). Topics include: software architecture, and software design patterns; maintenance and reverse engineering; quality assurance; verification, validation, testing, and analysis; formal methods and theories; component-based development and reuse; safety, security, and risk management; dependability and reliability; empirical studies, and industrial best practices; applications and tools; etc.

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

April 13-17 8th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2015), Graz, Austria. Deadline for submissions: October 6, 2014 (workshops), October 24, 2014 (research papers), January 16, 2015 (Ph.D. Symposium), February 16, 2015 (Testing Tools track), February 23, 2015 (Testing in Practice papers).

♦ April 20-24 17th **International Real-Time Ada Workshop** (IRTAW'2015), Vermont, New York, USA. In cooperation with AdaCore and Ada-Europe. Deadline for submissions: February 4, 2015 (position papers).

April 22-24 XVIII **Iberoamerican Conference on Software Engineering** (CIbSE'2015), Lima, Peru. Topics include: languages, methods, processes, and tools; reverse engineering and software system modernization; software evolution and maintenance; model-driven engineering; proof, verification, and validation; quality, measurement, and assessment of products and processes; formal methods applied to software engineering; software product families and variability; software reuse; reports on benefits derived from using specific software technologies; quality measurement; experience management; systematic reviews and evidence-based software engineering; industrial experience and case studies; etc. Deadline for submissions: December 8, 2014 (abstracts), December 15, 2014 (papers).

April 27-29 7th **NASA Formal Methods Symposium** (NFM'2015), Pasadena, California, USA. Topics include: identifying challenges and providing solutions to achieving assurance in mission- and safety-critical systems, model checking, static analysis, modeling and specification formalisms, model-based development, applications of formal methods to aerospace systems and cyber-physical systems, etc. Deadline for submissions: November 10, 2014 (papers).

April 29-30 10th **International Conference on Evaluation of Novel Approaches to Software Engineering** (ENASE'2015), Barcelona, Spain. Topics include: comparing novel approaches with established traditional practices and evaluating them against software quality criteria, software process improvement, model-driven engineering, application integration technologies, software quality management, software change and configuration management, geographically distributed software development environments, formal methods, component-based software engineering and commercial-off-the-shelf (COTS) systems, software and systems development methodologies, etc. Deadline for submissions: November 19, 2014 (papers), January 9, 2015 (position papers).

☺ May 16-24 37th **International Conference on Software Engineering** (ICSE'2015), Firenze, Italy. Topics include: component-based software engineering; debugging, fault localization, and repair; dependability, safety, and reliability; embedded and cyber physical systems; formal methods, verification, and synthesis; middleware, frameworks, and APIs; model-driven engineering; parallel, distributed, and concurrent systems; performance; program analysis; programming, specification, and modeling languages; reverse engineering; security, privacy and trust; software architecture; software economics, management, and metrics; software evolution and maintenance; software modeling and design; software product lines; software reuse; tools and environments; etc. Deadline for submissions: October 10, 2014 (workshop proposals, technical briefings proposals), October 24, 2014 (Joint SE Education and Training - JSEET, Software Engineering In Practice - SEIP, Software Engineering in Society - SEIS), November 21, 2014 (New Ideas and Emerging Results - NIER, doctoral symposium, ACM student research competition, demonstrations), November 30, 2014 (SCORE-it team registration), January 13, 2015 (posters), February 15, 2015 (SCORE-it deliverable submission).

May 25-29 29th IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2015), Hyderabad, India. Topics include: parallel and distributed algorithms, applications of parallel and distributed computing, parallel and distributed software, including parallel and multicore programming languages and compilers, runtime systems, parallel programming paradigms, programming environments and tools, etc. Deadline for submissions: October 10, 2014 (abstracts), October 17, 2014 (papers).

♦ June 22-26 20th **International Conference on Reliable Software Technologies - Ada-Europe'2015**, Madrid, Spain. Sponsored by Ada-Europe, in cooperation requested with ACM SIGAda, SIGBED, SIGPLAN, and the Ada Resource Association (ARA). Deadline for submissions: January 11, 2015 (papers, tutorials, workshops), January 25, 2015 (industrial presentations).

June 22-26 20th **International Symposium on Formal Methods** (FM'2015), Oslo, Norway. Topics include: interdisciplinary formal methods (techniques, tools and experiences demonstrating formal methods in interdisciplinary frameworks); formal methods in practice (industrial applications of formal methods, experience with introducing formal methods in industry, tool usage reports, etc); tools for formal methods (advances in automated verification and model-checking, integration of tools, environments for formal methods, etc); role of formal methods in software and systems engineering (development processes with formal methods, usage guidelines for formal methods, method integration, qualitative or quantitative improvements); theoretical foundations (all aspects of theory related to specification, verification, refinement, and static and dynamic analysis). Deadline for submissions: October 30, 2013 (hosting proposals), January 2, 2015 (abstracts), January 9, 2015 (full papers).

☺ Sep 01-04 **International Conference on Parallel Computing 2015** (ParCo'2015), Edinburgh, Scotland, UK. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments, in particular parallel programming languages, compilers, and environments, tools and techniques for generating reliable and efficient parallel code, testing and debugging techniques and tools, best practices of parallel computing on multicore, manycore, and stream processors, etc. Deadline for submissions: February 28, 2015 (extended abstracts), March 31, 2015 (mini-symposia).

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

Association for Computing Machinery

Advancing Computing as a Science & Profession

HILT 2014: HIGH INTEGRITY LANGUAGE TECHNOLOGY

ACM SIGAda's Annual International Conference

Co-Located with OOPSLA/SPLASH 2014

October 18–21, 2014 / Portland, Oregon / Advance Program

High integrity software must not only meet correctness and performance criteria but also satisfy stringent safety and/or security demands, typically entailing certification against a relevant standard.

A significant factor affecting whether and how such requirements are met is the chosen language technology and its supporting tools: not just the programming language(s) but also languages for expressing specifications, program properties, domain models, and other attributes of the software or overall system.

HILT 2014 will provide a forum for experts from academia/research, industry, and government to present their latest findings in designing, implementing, and using language technology for high integrity software.

HILT attendees are invited to attend the SPLASH opening keynote address by Gary McGraw, CTO of Cigital, Inc. and author of *Software Security*.

Sponsored by SIGAda, ACM's Special Interest Group on the Ada Programming Language, in cooperation with SIGAPP, SIGBED, SIGCAS, SIGCSE, SIGPLAN, SIGSOFT, Ada-Europe, and the Ada Resource Association.

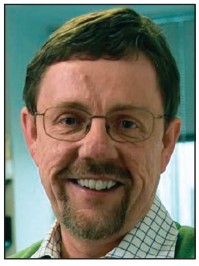
FEATURED SPEAKERS

A Decade of Program Verification at Microsoft

TOM BALL
Microsoft Research

From Ada 9X to Spaceport America: Going Where No One Has Gone Before

CHRISTINE ANDERSON
Spaceport America



AADL and Model-Based Engineering

PETER FEILER
Software Engineering Institute/
Carnegie Mellon University

CORPORATE SPONSORS

PLATINUM LEVEL

AdaCore
The GNAT Pro Company

GOLD LEVEL

Microsoft Research

SILVER LEVEL

Ellidiss Software
TNI Europe Limited

PRE-CONFERENCE TUTORIALS / October 18–19

Saturday Full Day / 9:00 AM–5:30 PM / SAT_FD_1
Ed Colbert (Absolute Software) / ***Object-Oriented Programming with Ada 2005 and Ada 2012***

Saturday Full Day / 9:00 AM–5:30 PM / SAT_FD_2
Peter Chapin (Vermont Technical College) and John W. McCormick (University of Northern Iowa) / ***Introduction to SPARK 2014***

Sunday Morning / 9:00 AM–12:30 PM / SUN_AM_1
Ben Brosgol (AdaCore) / ***High-Integrity Object-Oriented Programming with Ada 2012***

Sunday Afternoon / 2:00 PM–5:30 PM / SUN_PM_1
Jérôme Hugues (Institute for Space and Aeronautics Engineering) and Frank Singhoff (Université de Bretagne Occidentale)
AADLv2, an Architecture Description Language for the Analysis and Generation of Embedded Systems

Sunday Afternoon / 2:00 PM–5:30 PM / SUN_PM_2
Niko Matsakis (Mozilla Research) / ***Rust—Zero-Cost Safety***

TECHNICAL PROGRAM / October 20–21

MONDAY

9:00 AM–10:30 AM
Greetings
SIGAda and Conference Officers

Keynote Address
Christine Anderson (Spaceport America)
From Ada 9X to Spaceport America: Going Where No One Has Gone Before

10:30 AM–11:00 AM Break / Exhibits

11:00 AM–12:30 PM
Enhancing and Evolving Embedded Systems Languages for Safety

J. Barnes and T. Taft
Ada83 to Ada 2012—Lessons Learned Over 30 Years of Language Design

D. Crocker
Can C++ Be Made as Safe as SPARK?

T. Szabo
mbeddr—Extensible Languages for Embedded Software Development

Sponsor Presentation

12:30 PM–2:00 PM Break / Exhibits

2:00 PM–3:30 PM
Model-Based Engineering

Invited Talk
Peter Feiler (SEI/CMU)
AADL and Model-Based Engineering

A. Gacek
Resolute: An Assurance Case Language for Architecture Models

3:30 PM–4:00 PM Break / Exhibits

4:00 PM–5:30 PM
Behavioral Modeling and Code Generation

E. Ahmad
Hybrid Annex: An AADL Extension for Continuous Behavior and Cyber-Physical Interaction Modeling

J. Hugues
Leveraging Ada 2012 and SPARK 2014 for Assessing Generated Code from AADL Models

Industrial Presentations

B. Larson
BLESS Assertion as Behavioral Interface Specification Language

E. Seidewitz
UML with Meaning: Executable Modeling in Foundational UML and the Alf Action Language

Panel: Executable and Behavioral Modeling Languages
E. Ahmad, J. Hugues, B. Larson, E. Seidewitz

7:00 PM–10:00 PM
Dinner and Social Event

TUESDAY

8:30 AM–10:00 AM
Announcements
SIGAda Awards
Ricky E. Sward, Past SIGAda Chair

Keynote Address
Tom Ball (Microsoft Research)
A Decade of Program Verification at Microsoft

10:00 AM–10:30 AM Break / Exhibits

10:30 AM–12:30 PM
Applying Formal Methods

W. Rathje
A Framework for Model Checking UDP Network Programs with Java Pathfinder

A. H. Bagge
Specification of Generic APIs—or: Why Algebraic May Be Better than Pre/Post

Panel: Practical Use of Assertions and Formal Methods in Industry
I. Ahmed, *Formal Methods for Commercial Applications*; A. H. Bagge, *Algebraic API Specifications*; R. A. Reyes, *Knowledge Base for Use of Assertions*

12:30 PM–2:00 PM Break / Exhibits

2:00 PM–3:30 PM
Safe Programming Languages for the Multicore Era (I)

T. Taft
Safe Parallel Programming in Ada with Language Extensions

R. Bocchino
Spot: A Programming Language for Verified Flight Software

N. Matsakis
The Rust Langauge

Sponsor Presentation

3:30 PM–4:00 PM Break / Exhibits

4:00 PM–5:30 PM
Safe Programming Languages for the Multicore Era (II)

Panel: Finding Safety in Numbers—New Languages for Safe Multicore Programming and Modeling
R. Bocchino, N. Matsakis, T. Taft, B. Larson, E. Seidewitz

5:30 PM–6:00 PM
HILT 2014 Conference Wrap-up
SIGAda and Conference Officers

For more information and updates, visit www.sigada.org/conf/hilt2014

SPLASH OPENING KEYNOTE ADDRESS / October 22

8:30 AM–10:00 AM
HILT attendees are invited to attend the Wednesday morning SPLASH opening keynote address by Gary McGraw, CTO of Cigital, Inc. and author of *Software Security*

28TH GI/ITG International Conference on Architecture of Computing Systems
Porto, Portugal 24-27 March 2015

CALL FOR PAPERS

THIS YEAR'S FOCUS: Reconciling Parallelism and Predictability in Mixed-Critical Systems

www.cister.isep.ipp.pt/arcs2015

The ARCS series of conferences has over 30 years of tradition reporting high quality results in computer architecture and operating systems research. The focus of the 2015 conference will be on **Reconciling Parallelism and Predictability in Mixed-Critical Systems**. Like the previous conferences in this series, it continues to be an important forum for computer architecture research.

The proceedings of ARCS 2015 will be published in the Springer Lecture Notes on Computer Science (LNCS) series. After the conference, authors of selected papers will be invited to submit an extended version of their contribution for publication in a special issue of the Journal of Systems Architecture. Also, a best paper and best presentation award will be provided at the conference.

Authors are invited to submit original, unpublished research papers on one of the following topics:

- Multi-/many-core architectures, memory systems, and interconnection networks.
- Models and tools for multi-/many-core systems including but not limited to programming models, runtime systems, middleware, and verification.
- Design, methods, and hardware and software architectures for mixed-critical systems.
- Architectures and design methods/tools for robust, fault-tolerant, real-time embedded systems.
- Generic and application-specific accelerators in heterogeneous architectures.
- Cyber-physical systems and distributed computing architectures.
- Adaptive system architectures such as reconfigurable systems in hardware and software.
- Organic and Autonomic Computing including both theoretical and practical results on self-organization, self-configuration, self-optimization, self-healing, and self-protection techniques.
- Operating Systems including but not limited to scheduling, memory management, power management, and RTOS.
- Energy-awareness and green computing.
- System aspects of ubiquitous and pervasive computing such as sensor nodes, novel input/output devices, novel computing platforms, architecture modeling, and middleware.
- Grid and cloud computing.

Submissions

Regular papers should be submitted via the link provided on the conference website, formatted according to the Springer LNCS style and not exceeding 12 pages.

Workshop and Tutorial proposals within the technical scope of the conference are solicited. Those should be submitted by email directly to the corresponding chair (address at the website).

Important Dates

Paper submission deadline:	**October 6, 2014**
Workshop/tutorial proposals:	**November 3, 2014**
Notification of acceptance:	**December 1, 2014**
Camera-ready papers:	**December 15, 2014**

Organizing Committee

General Co-Chairs
Luís Miguel Pinho, CISTER/INESC-TEC, ISEP, Portugal
Wolfgang Karl, Karlsruhe Institute of Technology, Germany

Program Co-Chairs
Albert Cohen, INRIA, France
Uwe Brinkschulte, Universität Frankfurt, Germany

Workshop and Tutorial Co-Chair
João Cardoso, FEUP/University of Porto, Portugal

Industrial Liaison Co-Chairs
Sascha Uhrig, Technische Universität Dortmund, Germany
David Pereira, CISTER/INESC-TEC, ISEP, Portugal

Poster Co-Chairs
Kluge Florian, University of Augsburg, Germany
Patrick Meumeu Yomsi, CISTER/INESC-TEC, ISEP, Portugal

Publicity Chair
Vincent Nelis, CISTER, ISEP, Portugal

Publication Chair
Thilo Pionteck, Hamburg University of Technology, Germany

Local Organization Chair
Luis Ferreira, CISTER/INESC-TEC, ISEP, Portugal

17th International Real-Time Ada Workshop - IRTAW 2015

in cooperation with AdaCore and Ada-Europe

www.cs.york.ac.uk/~andy/IRTAW2015

Vermont, USA

Week of 20-24 April 2015 (actual dates TBD)

Call for Papers

Since the late Eighties 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 IRTAW meetings have significantly contributed to the Ada 2005 and Ada 2012 standards, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar profile.

In keeping with this tradition, the goals of IRTAW-17 will be to:

- review the current status of the Ada 2012 Issues that are related with the support of real-time systems;
- examine experiences in using Ada for the development of real-time systems and applications, especially – but not exclusively – those using concrete implementation of the new Ada 2012 real-time features;
- report on or illustrate implementation approaches for the real-time features of Ada 2012;
- consider the added value of developing other real-time Ada profiles in addition to the Ravenscar profile;
- examine the implications to Ada of the growing use of multiprocessors in the development of real-time systems, particularly with regard to predictability, robustness, and other extra-functional concerns;
- examine and develop paradigms for using Ada for real-time distributed systems, with special emphasis on robustness as well as hard, flexible and application-defined scheduling;
- consider the definition of specific patterns and libraries for real-time systems development in Ada;
- identify how Ada relates to the certification of safety-critical and/or security-critical real-time systems;

- examine the status of the Real-Time Specification for Java and other languages for real-time systems development, and consider user experience with current implementations and with issues of interoperability with Ada in embedded real-time systems;
- consider the lessons learned from industrial experience with Ada and the Ravenscar Profile in actual 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 IRTAW-17 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, however, be given to those submitting 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 under consideration at Ada-Europe (deadline 11th January, 2015) may offer an extended abstract of the same material to IRTAW-17.

Please submit position papers, in PDF, to the Program Chair by e-mail: andy.wellings@york.ac.uk

Important Dates

- Paper Submission: **4 February, 2015**
- Notification of Acceptance: 1 March, 2015
- Confirmation of Attendance: 14 March, 2015
- Final Paper Due: 1 April, 2015
- Workshop: April TBD in week of 20-24, 2015

Program Chair

- Andy Wellings, University of York

Workshop Chair

- Robert Dewar, AdaCore

Program Committee Members

Mario Aldea Rivas, John Barnes, Ben Brosgol, Alan Burns, Michael Gonzàlez Harbour, José Javier Gutiérrez, Stephen Michell, Brad Moore, Luís Miguel Pinho, Juan Antonio de la Puente, Jorge Real, Jose F. Ruiz, Joyce Tokar, Tullio Vardanega, Andy Wellings and Rod White.

Call for Papers

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

22-26 June 2015, Madrid, Spain

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

Conference Chair

Alejandro Alonso
ETSIT-UPM
alonso@dit.upm.es

Program co-Chairs

Juan A. de la Puente
ETSIT-UPM
jpuente@dit.upm.es

Tullio Vardanega
Università di Padova
tullio.vardanega@unipd.it

Tutorial Chair

Jorge Real
UPV
jorge@disca.upv.es

Exhibition Chair

Santiago Urueña
GMV
suruena@gmv.com

Industrial Chair

Jørgen Bundgaard
Rambøll Danmark A/S
jogb@ramboll.dk

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

Publicity Chair

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

Local Chair

Juan Zamorano
ETSIINF-UPM
jzamora@fi.upm.es

"In cooperation" requested with
ACM SIGAda, SIGBED, SIGPLAN, and ARA

Program Committee

General Information

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

Schedule

11 January 2015	Submission of regular papers, tutorial and workshop proposals
25 January 2015	Submission of industrial presentation proposals
1 March 2015	Notification of acceptance to all authors
29 March 2015	Camera-ready version of regular papers required
12 April 2015	Industrial presentations abstracts required
17 May 2015	Tutorial and workshop materials required

Topics

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

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

- **Multicore and Manycore Programming**: Predictable Programming Approaches for Multicore and Manycore Systems, Parallel Programming Models, Scheduling Analysis Techniques.
- **Real-Time and Embedded Systems**: Real-Time Scheduling, Design Methods and Techniques, Architecture Modelling, HW/SW Co-Design, Reliability and Performance Analysis.
- **Mixed-Criticality Systems**: Scheduling methods, Mixed-Criticality Architectures, Design Methods, Analysis Methods.
- **Theory and Practice of High-Integrity Systems**: Medium to Large-Scale Distribution, Fault Tolerance, Security, Reliability, Trust and Safety, Languages Vulnerabilities.
- **Software Architectures**: Design Patterns, Frameworks, Architecture-Centred Development, Component-based Design and Development.
- **Methods and Techniques for Software Development and Maintenance**: Requirements Engineering, Model-driven Architecture and Engineering, Formal Methods, Re-engineering and Reverse Engineering, Reuse, Software Management Issues, Compilers, Libraries, Support Tools.
- **Software Quality:** Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems.
- **Mainstream and Emerging Applications**: Manufacturing, Robotics, Avionics, Space, Health Care, Transportation, Cloud Environments, Smart Energy systems, Serious Games, etc.
- **Experience Reports in Reliable System Development**: Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics.
- **Experiences with Ada and its Future**: Reviews of the Ada 2012 new language features, implementation and use issues, positioning in the market and in the software engineering curriculum, lessons learned on Ada Education and Training Activities with bearing on any of the conference topics.

Call for Regular Papers

Authors of regular papers which are to undergo peer review for acceptance are invited to submit original contributions. Paper submissions shall not exceed 14 LNCS-style pages in length. Authors shall submit their work via EasyChair following the relevant link on the conference web site. The format for submission is solely PDF.

Proceedings

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

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

Awards

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

Call for Industrial Presentations

The conference seeks industrial presentations which deliver value and insight but may not fit the selection process for regular papers. Authors are invited to submit a presentation outline of exactly 1 page in length by January 25, 2015. Submissions shall be made via EasyChair following the relevant link on the conference web site. The Industrial Committee will review the submissions and make the selection. The authors of selected presentations shall prepare a final short abstract and submit it by April 12, 2015, aiming at a 20-minute talk. The authors of accepted presentations will be invited to submit corresponding articles for publication in the *Ada User Journal (http://www.ada-europe.org/auj/)*, which will host the proceedings of the Industrial Program of the Conference. For any further information please contact the Industrial Chair directly.

Call for Tutorials

Tutorials should address subjects that fall within the scope of the conference and may be proposed as either half- or full-day events. Proposals should include a title, an abstract, a description of the topic, a detailed outline of the presentation, a description of the presenter's lecturing expertise in general and with the proposed topic in particular, the proposed duration (half day or full day), the intended level of the tutorial (introductory, intermediate, or advanced), the recommended audience experience and background, and a statement of the reasons for attending. Proposals should be submitted by e-mail to the Tutorial Chair. The authors of accepted full-day tutorials will receive a complimentary conference registration as well as a fee for every paying participant in excess of 5; for half-day tutorials, these benefits will be accordingly halved. The *Ada User Journal* will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

Workshops on themes that fall within the conference scope may be proposed. Proposals may be submitted for half- or full-day events, to be scheduled at either end of the conference.

Call for Workshops

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

Call for Exhibitors

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

Grants for Reduced Student Fees

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

Critical Software for the First European Rail Traffic Management System

Ana Rodríguez
SILVER ATENA Spain, Ronda de Poniente 5, 28760 Tres Cantos, Madrid, Tel: +34 608754488; email: ana.rodriguez@silver-atena.es

Abstract

SILVER ATENA participates in various projects on Advance Traffic Management & Control (ATMC) Systems which aim to develop a new generation of signalling and control systems, building on current European Rail Traffic Management System (ERTMS), to enable intelligent traffic management with automatically driven trains and optimise capacity, reliability and minimise life-cycle cost.

The Radio Block Centre (RBC) of the ERTMS trackside sub-system is a computer-based system that elaborates messages to be sent to the train on basis of information received from external trackside systems and on basis of information exchanged with the on-board sub-systems. The RBC software is implemented in Ada Language.

Keywords: safety-critical, ERTMS, Railways Industry

1 Introduction

Harmonized Community transport policies are essential in a European union of 28 countries. The ability to circulate from one member state to other using interoperable driving systems has became a fundamental requirement.

Achieving a single automatic driving system is crucial for the optimization of rail transport efficiency on a European scale. In order to produce such a system, it is necessary to establish common standards for on-board systems, the connection/communication interfaces between modules and the development of common procedures. In order to fulfil these requirements the ERTMS [1] has been developed and it is now being deployed across Europe.

Since 2009, SILVER ATENA [2] collaborates with Siemens Rail Automation Division [3] (former Invensys Rail Dimetronic) in the development of the Radio Block Centre (RBC) that implements ERTMS interoperability requirements that are mainly related to the data exchange between the RBC and the on-board sub-system. SILVER ATENA also collaborates with Bombardier [4] on the improvement of Rio de Janeiro commuter lines that is the first ERTMS solution deployed in South America.

The European Rail Industry is now doing an unprecedented joint effort to massively enhance the capacity of the European rail system in order to cope with increased passenger and freight demand as a result of societal pressures in support of green transport and elicit a step-change in the reliability of next generation products and solutions while reducing their life cycle costs. It aims to attract passengers and businesses to rail transport and increase the competitiveness of the European rail industry vis-à-vis emerging Asian competition.

2 The European Rail Traffic Management System

The European Rail Traffic Management System/ European Train Control System (ERTMS/ETCS) is the European system for automatically controlling train movement and is the solution identified by Europe's railways and industries for achieving rail interoperability.

Following the decision taken by the transport ministers in December 1989 a group of railway experts began studying the fundamental requirements for a European interoperable system and defining the Project: the UIC-ERRI-S 1069 Declaration elaborated by a study group from the Union Internationale des Chemins de Fer (UIC)/ the International Union of Railways [5].

In June 1991, the industry and the railways (UIC) agreed on the principles for pursuing collaboration to develop new equipment for the European control and command system based on the ERRI-UIC specifications.

After the definition of the ERTMS/ETCS Master Plan by the Commission, in 1999, the French, German, Italian, Spanish, British and Dutch railways jointly started the European Economic Interest Group ERTMS/ETCS Users Group (EEIG-ERTMS/ETCS Users Group).

Since then, the railway administrations and the industries worked together within AEIF (European Association for Railway Interoperability) to develop the essential requirements, the interoperability components and interfaces, the European procedures for evaluating conformity/suitability of use, the functional and technical specifications, and the migration strategy for interoperability.

A great effort is currently being deployed to harmonise the overall migration plans of the European networks. "Core network corridors" were introduced to facilitate the coordinated implementation of the core network, also aiming at integrating rail freight corridors, promoting clean fuel and other innovative transport solutions, advancing Telematics applications for efficient infrastructure use, integrating urban areas into the TEN-T, enhancing safety.

Nine core network corridors are identified in the annex to the CEF Regulation, which includes a list of projects pre-identified for possible EU funding during the period 2014 – 2020, based on their added value for TEN-T (EU funding during the period 2014 – 2020) development and their maturity status.

A map of the corridors [6] recently agreed upon is shown below.

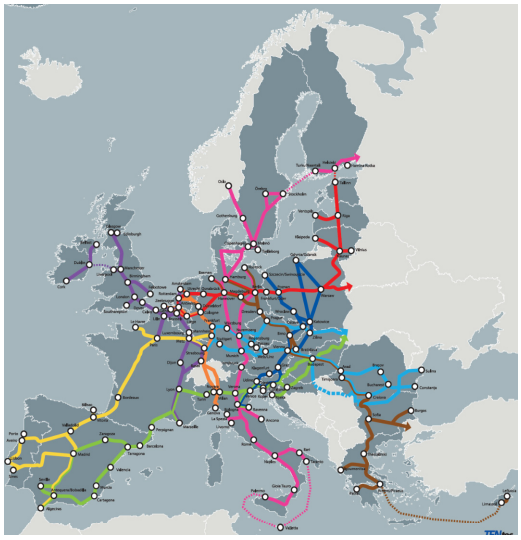


Figure 1 European Commission – Trans-European Transport Network

The specifications of ERTMS/ETCS requirements are public, and define the so-called kernel and its interfaces with the ground.

SILVER ATENA is devoted to assist our clients in the development and prove of ERTMS/ETCS equipments, which is intended to achieve this interoperability *with safety*.

3 Levels of application

The ERTMS/ETCS system provides the driver, in a standard format, with all the information needed for optimum driving, constantly controlling the effect of every action taken in terms of train safety, and activating emergency braking should the train speed exceed the maximum safety limits

There are three levels of application that define the ways in which the train can receive lineside information, as illustrated in the following figure.

Table 1 ERTMS/ETCS equipment

	On board	Track-side				
ERTMS/ETCS level	Check of train integrity	Data transmission	Lineside electronic units	Lineside signals	Track occupancy detection	Radioblock
1	no	balises+loops *(option)*	yes	yes	yes	no
2	no	balises+radio	no	no	yes	yes
3 *(planned)*	yes	balises+radio	no	no	no	yes

Figure 2 ERTNS/ETCS Levels of applications

The Application level 2 uses a continuous type of communication system for the onboard transmission of lineside data through safe radio links between a Radio Block Centre (RBC) and the train.

The information in the radio messages and their coding method and allocation in the telegram transmitted are standards defined in the ERTMS/ETCS specifications.

Also for ERTMS/ETCS level 2, the position of the trains is determined by the conventional position detection systems (track circuits) while the onboard software manages the functionality using available ground-based data and train-borne data.

4 The Radio Block Center

The ERTMS/ETCS level 2's ground system comprises a RBC central unit, installed in specific central posts, from which railway circulation is managed and controlled through the System of Command and Control (SCC).

4.1 ERTMS/ETCS level 2's ground technology

The RBC continuously transmits to every train, via GSM-R (Global System for Mobile Communications – Railway), the speed and the train distance to be observed depending on the position of all the trains present on the line (train distancing) and the constraints imposed by the track At the same time, the trains send out radio signals indicating their position to the central unit.

At the same time, the trains send out radio signals indicating their position to the central unit (ERTMS Functional Specification [7] and [8]). On the basis of the state of the infrastructure (free line, routes in the stations, train speeds, slowdowns) and the position of the train, the RBC transmits "movement authority" data to the on-board unit, giving details of the free distance and the maximum permitted speed at the point.

The environment of ERTMS/ETCS system (Figure 3) is composed of:

- The train, which will then be considered in the train interface specification;
- The driver, which will then be considered via the driver interface specification;
- Other onboard interfaces, and
- External trackside systems (interlocking, control centres, etc.), for which no interoperability requirement will be established.

4.2 System Operation

Commercial solutions for ERTMS/ETCS systems compete in providing optimal performance for system operation as well as enhanced maintenance features for the railways systems administration.

The architecture of the RBC system includes all those elements that improve operability (safe and reliable functioning condition), modularity (that facilitates maintenance and improves reliability) and connectivity (thought standard interfaces) of a safety-critical system.

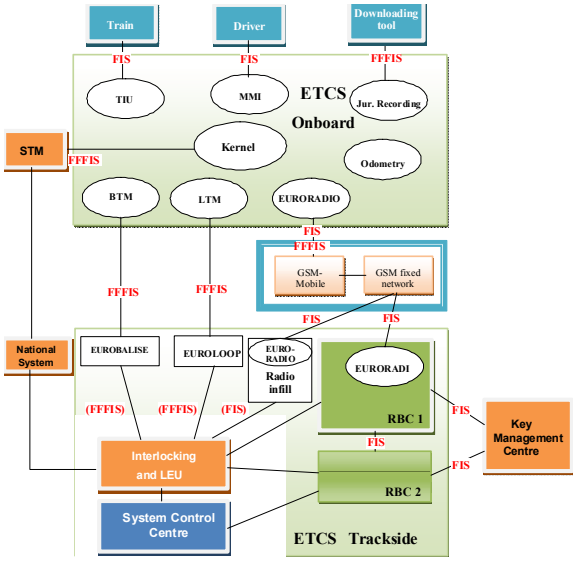


Figure 3 ERTMS/ETCS System [8]

The environment of RBC system (Figure 4) is composed of:

- Adjacent RBC: bi-directional continuous information by GSM-R Euro-radio (EN 50129 SIL4 Ada Software)
- ETCS on-board: bi-directional continuous information by GSM-R Euro-radio (EN 50129 SIL4 Ada Software)
- CEC- Command and control of all the RBCs in a line
- JRU – black-box unit
- Maintenance Assistance Unit
- I/C Control equipment
- Local ERTMS Control, operator commands console

SILVER ATENA had an important role in those system features concerning RAMS (Reliability, Availability, Maintainability and Safety).

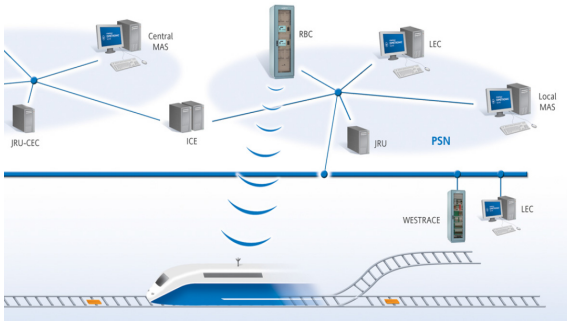


Figure 4 RBC System [9]

4.3 Safety Integrity Requirements

The Safety Integrity Level (SIL) is a widespread concept used all over the industries which deal with the development of safety-related systems.

The EN 50126-1 standard [11], defines the term "Safety Integrity" as: "The likelihood of a system satisfactorily performing the required safety functions under all the stated conditions within a stated period of time". This standard also defines the level of safety integrity as: "One of a number of defined discrete levels for specifying the safety integrity requirements of the safety functions to be allocated to the safety related systems".

The process for allocating a SIL to a safety function starts by performing a risk analysis, in order to identify and quantify the hazards that a failure of our system may cause. The risk analysis establishes the measures which shall be taken so that the risk can be reduced to an acceptable level.

Figure 5 Risk Analysis

As discussed in [10], the concept of SIL is essentially linked to the safety functions. Consequently, it is fundamental to carry out a risk analysis, which will enable us to obtain the safety functions of our system. Based on those safety functions, a SIL is allocated, considering also the risk reduction factor: the higher the risk, the higher the SIL.

The RBC system must comply with a SIL 4 safety integrity level mainly linked to its messages generation function that must maintain and assure the accuracy, consistency and validity of data.

The life-cycle of the RBC system must accomplish a strict life-cycle development to eliminate (minimize) threats to data integrity. Following UNE-EN 50129 [13] and IEC 61508 [14] standards, the following four conditions shall be satisfied in order to comply with a given SIL:

- Quality management
- Safety management
- Functional and technical safety
- Quantified safety targets

The normative also mandates independent development teams: Design& Development team, Verification and Validation (V&V) team and a safety auditor to provide an Independent Safety Assessment (ISA).

The RBC design methodologies conforms the European standards EN 50126-1 [11], EN 50128 [12] and EN 50129 [13]. EN 50128 safety process is illustrated in the figure below.

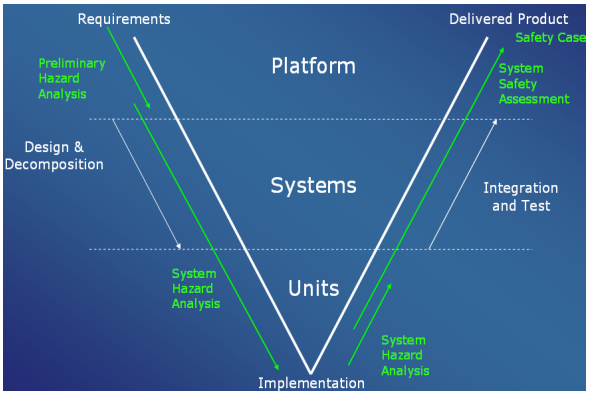


Figure 6 EN 50128 Safety Process

The design of a safety critical system shall incorporate controls and fault-tolerance techniques to eliminate or reduce the probability or severity of each hazard, to lower the overall risk. The RBC system incorporates a two out of three voting system and their communication equipments are design in a hot stand-by configuration.

4.4 Software Validation Development

The RBC embedded software falls into the safety-critical category, i.e., software which can directly create or control a hazard and that provides information required for a safety-related decision falls into the safety-critical category

One of the most important aspects of a software design for a safety-critical system is designing for minimum risk. This "minimum risk" includes hazard risk (likelihood and severity), risk of software defects, risk of human operator errors, and other types of risk (such as programmatic, cost, schedule, etc.).

SILVER ATENA has significantly contributed to the design, specification and implementation of the RBC V&V Program with include the system functional testing and the integration and performance testing. The V&V program eventually has aimed to demonstrate that the verification and validation activities performed for critical software are appropriate and sufficient to fulfil the safety and effectiveness requirements for the system.

5 Challenges and Opportunities

Currently Spain has become the ERTMS/ETCS largest deployment in Europe, offering a significant profile of consolidated technologies and capacities. There are excellent opportunities for Spanish railways industry (e.g. AVE La Meca-Medina) and the market is expected to keep on growing at a healthy at ~7%.

New challenges for industry are facing the deployment of ERTMS/ETCS level 2 and 3 and related regulations as well as measures aim to strength and improve safety.

The introduction of satellite assets in conjunction with existing terrestrial railways /systems is an important challenge as well. Satellites increase the viability of ERTMS for low-traffic lines by avoiding the need for expensive track equipment and dedicated telecom networks. Virtual beacons could be used instead and the train's position is fixed by Satellite Navigation (SatNav).

Currently Industry demands improvements of the safety assurance processes, for both the deployment and the operations of the train lines.

Although no programming language is guaranteed to produce safe software, Ada is widely used for railways critical developments (SIL4 and SIL3) because it enforces good programming practices, makes bugs easier for the compiler to find, and incorporates elements that make the software easier to verify.

The "safeness" and reliability of a system depend on many factors: Humans are involved in all aspects of the process, quite capable of subverting even the "safest" of languages. The Santiago de Compostela (Spain) derailment occurred in July 2013, when a high-speed train travelling from Madrid, derailed at high speed on a curve, revealed that no ERTMS were installed at site. An expert report commissioned by the Spanish government (June 2014) conclude that the cause of the crash was "excess speed resulting from the driving personnel's failure to comply with speed limit regulations. Recommendations claimed for improving the ERTMS signs warning drivers and security mechanisms to automatically slow down speeding trains, and a safer internal communications system.

References

[1] The European Rail Traffic Management System; http://www.ertms.net/

[2] SILVER ATENA http://www.silver-atena.com

[3] Siemens Rail Systems Division; http://www.siemens.com/about/en/businesses/infrastructure_and_cities/rail_systems.htm

[4] Bombardier Transportation; http://www.bombardier.com/en/transportation.html

[5] The International Union of Railways http://www.uic.org/

[6] European Commission Mobility and Transport (2014) http://ec.europa.eu/transport/themes/infrastructure/ten-t-guidelines/

[7] UNISIG (2007), *FRS V5.00, Functional System Requirements Specification*, ERTMS Users Group.

[8] UNISIG (2010), *SRS Subset026 V2.3.0, System Requirements Specification*, ERTMS Users Group.

[9] Siemens (former Invensys) Rail Systems, ERTMS FUTUR 2500 - ERTMS Level 2, (2011).

[10] SILVER ATENA (2013), *Newsletter White Paper: The controversial SIL.*

[11] UNE-EN 50126-1 (2005), *Railway applications – The specification and demonstration of Reliability, Availability, Maintainability and Safety* (RAMS), (Including CORR: 2006 y CORR: 2010).

[12] UNE-EN 128 (2011), *Railway Applications – Communications, signalling and processing systems – Software for Railway control and protection systems*.

[13] UNE-EN 50129 (2005), *Railway Applications – Communications, signalling and processing systems - Safety related electronic systems for signalling*, (Including CORR: 2010).

[14] IEC 61508 (2008), *Functional safety of electrical/ electronic/programmable electronic safety-related systems*.

Privacy Leaks in Java Classes

Jacob Sparre Andersen
JSA Research & Innovation, Jægerparken 5, 2. th., 2970 Hørsholm, Denmark; Phone: +45 21 49 08 04; E-mail: jacob@jacob-sparre.dk

Abstract

One of the changes from DO-178B to DO-178C is supposedly to make it allowable to implement flight control systems in the Java programming language [3]. This makes it important to learn of the reliability deficiencies of Java, and possibly how to avoid them.

This paper is focused on one particular class of programming errors in Java; **privacy leaks** *from encapsulated data structures.*

This mistake seems to be quite common. Even the "getter" generator in the popular Java IDE Eclipse [2] makes the mistake for the programmers. This motivated me to put together a tool for identifying privacy leaking "getters" in Java classes. The tool uses a Java decompiler to generate a normalised form of the source code for the Java classes, and then uses simple pattern matching to identify uses of the simple type "getter" pattern for composite, private attributes.

One of the five goals in the creation of the Java programming language is that it should be "robust and secure". The extent of privacy leaks in real-life Java classes indicates that there is still quite a way to that goal.

1 Introduction

In Java assignment (=) works differently for simple and composite types (classes); simple type objects are copied while composite types have their reference copied.

One of the consequences of this design decision in Java is that "getters" for composite type attributes have to be written differently than those for simple type attributes. If one uses the simple type "getter" pattern for a composite type attribute, the result will be a privacy leak, where the user of the class will have access to modify a private attribute of the class directly – something which is in contrast with the whole idea of private attributes.

As an external examinator for Danish software engineering schools, I noticed that it seems to be difficult for students to grasp the practical consequences of the different handling of simple and composite types in Java. This has inspired me to look into how widespread privacy leaks are in "industrial" Java software, which again has lead to the development of the tool presented in this paper.

This paper will

- discuss potential bugs derived from privacy leaks;
- show examples of safe and privacy leaking "getters";
- document how Eclipse generates unsafe source code;
- show the pattern searched for by the privacy leak tracker; and
- finally discuss why I don't consider this a similarly serious issue in Ada.

2 Definition and consequences

First of all we need a precise definition of what a **privacy leak** is. In general a **privacy leak** can be defined as:

> When somebody "outside" gets a copy of an object meant to be securely "inside"... [1]

This definition is rather broad. More specifically, we are looking for cases where a client of a Java class can **modify** the (referenced) value of a **private attribute** of the class, because the class hands out references to its internal state [1].

When a class is implemented with private attributes, it can be to:

- Enforce a consistent state inside an object of that class.
- Make objects of that class immutable.

A privacy leak will break these possible intended features of making attributes private.

An "interesting" side-effect could for example be that a client accidentally modifies encryption parameters for an encrypted network connection, making the resulting encryption trivial to break. In general the side-effect is that the class can end up in an inconsistent state.

2.1 A safe "getter"

As an example, we write a class containing a counter with methods for

- incrementing the counter, and
- reporting the current value of the counter to standard output:

[1] The example in [1] goes the other way: A client passes a reference to an object to the class, and the class stores the reference instead of copying the object first.

```
public class NonLeaking {
   private Counter leakedCounter;

   public NonLeaking() {
      super();
      leakedCounter = new Counter();
   }

   public void increment() {
      this.leakedCounter.increment();
   }

   public void reportState() {
      System.out.println("NonLeaking:");
      this.leakedCounter.reportState();
   }
```

We want to be able to access the counter from clients of the class, so we introduce a "getter":

```
   /* Getter written by hand: */
   public Counter getLeakedCounter() {
      return new Counter(leakedCounter);
   }
```

Here the **new** operator creates a new object of class `Counter` as a copy of the private attribute `leakedCounter`.

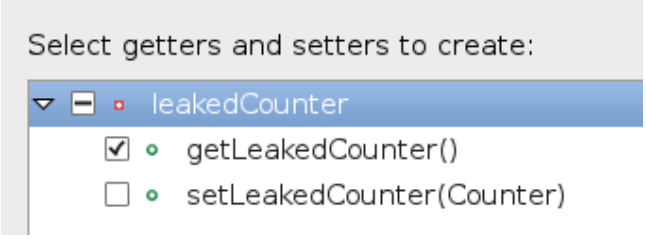
2.2 Generating a "getter" in Eclipse

We make a copy of the `NonLeaking` class without the `getLeakedCounter` method, and load it into Eclipse:

Counter.java | Leaking.java

```
package dk.jacob_sparre.privacy_leaks;

public class Leaking {
    private Counter leakedCounter;

    public Leaking() {
        super();
        leakedCounter = new Counter();
    }

    public void increment() {
        this.leakedCounter.increment();
    }

    public void reportState() {
        System.out.println("Leaking:");
        this.leakedCounter.reportState();
    }

}
```

We ask Eclipse to generate a "getter" for `leakedCounter`:



The result is:

Counter.java | *Leaking.java

```
package dk.jacob_sparre.privacy_leaks;

public class Leaking {
    private Counter leakedCounter;

    public Leaking() {
        super();
        leakedCounter = new Counter();
    }

    public void increment() {
        this.leakedCounter.increment();
    }

    public void reportState() {
        System.out.println("Leaking:");
        this.leakedCounter.reportState();
    }

    public Counter getLeakedCounter() {
        return leakedCounter;
    }

}
```

Notice how the "getter" generated by Eclipse simply returns a copy of the **reference** to the private attribute; giving the client access to modify the internal state of an object in the class directly.

2.3 Run-time behaviour

Having the two classes with respectively a hand-written and a generated "getter", we test them using this program:

```
public class Demo {
   public static void main(String[] args) {
      Leaking leaking = new Leaking();
      NonLeaking nonLeaking = new NonLeaking();

      leaking.reportState();
      leaking.increment(); // This should change
          the internal state of leaking.
      leaking.reportState();
      leaking.getLeakedCounter().increment(); //
          This shouldn't change the internal state
          of leaking.
      leaking.reportState();

      nonLeaking.reportState();
      nonLeaking.increment(); // This should change
           the internal state of leaking.
      nonLeaking.reportState();
      nonLeaking.getLeakedCounter().increment(); //
           This shouldn't change the internal state
           of leaking.
      nonLeaking.reportState();
   }

}
```

Running the program gives this output:

```
Leaking:
Counter value: 0
Leaking:
Counter value: 1
```

```
Leaking:
Counter value: 2
NonLeaking:
Counter value: 0
NonLeaking:
Counter value: 1
NonLeaking:
Counter value: 1
```

We can see how the `Leaking` class leaks the `leakedCounter` allowing the client to increment the **internal value**.

3 The pattern of a privacy leaking "getter"

Step one is to identify private attributes. Only these are subject to the kind of privacy leaks we are looking for.

In the normalised sources files we work on, this reduces to identifying lines of the form:

["`final`"] "`private`" *type_identifier* *attribute_identifier* "`;`"

We are not interested in those of the private attributes, which are immutable, or by-copy (i.e. simple) types. The attributes whose types are known to be immutable or by-copy are removed from the list of found private attributes.

Step two is to identify cases where a method returns a simple copy of an attribute. These are the source of the kind of privacy leaks we are looking for.

In the normalised sources files we work on, this reduces to identifying lines of the form:

"`return this.`"*attribute_identifier* "`;`"

where the attribute is on our list from step one.

In case step two finds matching lines (i.e. potentially leaking "getters"), step three is to see if the class of the potentially leaked attribute is actually immutable, as this makes a simple "getter" safe.

If the class isn't already known to be mutable, a manual inspection of the class is required at this point.

Any "getters" which match after this check must be considered privacy leaking, and thus unsafe.

4 Conclusion

4.1 Tool results

The tool was tested on real-life Java classes used for high-integrity, security critical tasks (as a stand-in for safety-critical Java classes). The tool was able to identify cases of privacy-leaks in the example classes, but definite security breaches were not identified.

The Eclipse IDE generates unsafe "getters" for by-reference types. Due to the internal structure of Eclipse, the generated "getters" can not be controlled by the type of the attribute in question. It might be worthwhile to substitute the templates in Eclipse such that the default generated "getter" uses **new**, and the programmer has to work to adapt the "getter" for by-copy and immutable types.

4.2 Comparison with Ada

The most relevant difference between Ada and Java with respect to the kind of privacy leaks presented here is:

- Assignment in Java works differently for simple and composite objects ("classes"), whereas Ada has consistent assignments (but requires the programmer worry about object or reference to object).

This means that a **superficially equivalent** record type and "getter" written in Ada will be **safe** with regard to privacy leaking.

Ada does have to option of storing references to attributes – as Java does it behind the scenes – but has to be done explicitly by the programmer.

It is my impression that the explicitness required by Ada is enough to make programmers aware of how the attribute should be handled, but it is not something I have documentation for.

The full source code for the demonstration example used in this paper can be downloaded from `http://repositories.jacob-sparre.dk/privacy-leaks-examples/`.

References

[1] CSE1030 – Introduction to Computer Science II – Lecture #8 – Aggregation & Composition II (2012).

[2] Eclipse Foundation Inc. (2014), Eclipse - The Eclipse Foundation open source community website.

[3] Guy L. Steele Jr. James Gosling, Bill Joy and Gilad Bracha (2005), *The Java Language Specification*, Addison-Wesley, 3rd edition.

Implementation of Task Types in AVR-Ada

André de Matos Pedro
CISTER/INESC TEC, ISEP, Polytechnic Institute of Porto, Portugal; anmap@isep.ipp.pt

Abstract

Ada contains a vast set of features, when supported in the runtime system, allowing a great flexibility for supporting memory management, exception handling, and task assignment in operating systems. Nevertheless, for embedded architectures many runtime features are not officially supported, either due to difficulties in the port, or due to the overheads generated by the runtime library. Therefore, the Zero Footprint approach is usually adopted to constraint Ada into a restricted subset that does not use any runtime feature, which precludes using some of Ada advantages over other languages. This work extends the current runtime library of AVR-Ada to support Ada task types for the AVR platform as have been done for other processor families, and describes an example of its usage using the Arducc board.

Keywords: Coroutines, tasks, AVR, Ada.

1 Introduction

The AVR family [1,2] is well known for its very low cost 8-bit and 32-bit micro-controller units (MCUs) widely used in a variety of embedded systems, such as water heater controllers, microwave and oven devices, brushless and brushed motor controllers, and several aeromodel controlling devices. A large number of open-hardware boards using AVRs such as Arduino [3], Ardupilot [4], and Ardusat [5] became very popular in the development of low-to-medium complexity embedded products.

Although there is support for AVR development using Ada, there is currently no runtime which supports the tasking model of Ada. For development with Ada, there are some references mainly for AVR32 micro-controllers [6]. For 8-bit AVR's we have a deterministic environment for Ada 2005, and methods for developing systems targeting Arduino boards [7] using the AVR-Ada cross compiler. Open-Source compiler options for programming AVR micro-controllers in Ada are: AVR-Ada [8] and GNAT AVR GPL [9]. To support development in Ada, there are two main tools: Eclipse [10] and GPS [11]. The Eclipse plugin [12] contains support to automatically upload and set different settings for AVR chips, while GPS supports the setting of compile properties for several chips using the GNAT project file.

In this paper we describe how to implement and use task types in Ada for 8-bit AVR micro-controllers extending the AVR-Ada runtime library. The tasks are coroutines [13] implemented using a set of assembly instructions and supported by 8-bit AVR devices. Furthermore, we point some preliminary results to support periodic hard real-time tasks as commonly used in several RTOS.

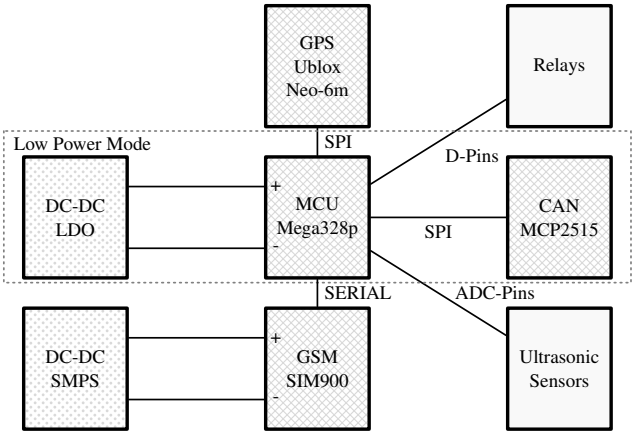


Figure 1: Arducc Architecture

Section 2 describes the used board, language, and tools. Section 3 describes how the task expansion is done in Ada 2012 and its respective dependencies, how coroutines and scheduler are related with task types, and how task types are mapped into AVR-Ada. Section 4 describes an example using Ada task types in AVR-Ada executing in our Arducc board. Lastly, Section 5 draws some conclusions and address further work.

2 Platform and Programs

2.1 Arducc: An AVR-based machine for lightweight automotive systems

Arducc [14] is an avr-based platform that is optimized for automotive and industrial systems. It supports the CAN bus protocol, includes a GSM modem for tracking purposes, a set of relays for general purpose, a low quiescent current voltage regulator for low current drain of MCU when battery powered, and a switched mode power supply (SMPS) for modem. MCU is an ATMega328P device with several capabilities such as low power AVR 8-Bit micro-controller, three timers, six pulse with modulation (PWM) channels, I2C and SPI bus, wake-up interrupt to wake on CAN, and 8 analog digital converter (ADC) channels.

A simplified architecture of the platform is shown in Figure 1. It has a low consumption segment, and contains two different power supplies allowing the board to have very low current consumption. In this setting, the MCU is switched to power down state, the CAN transceiver to sleep mode, and the SMPS is disabled. The system wakes-up and turns-on the SMPS and the remaining devices when one message is received from the CAN channel. The board layout can be seen in the Figure 2.

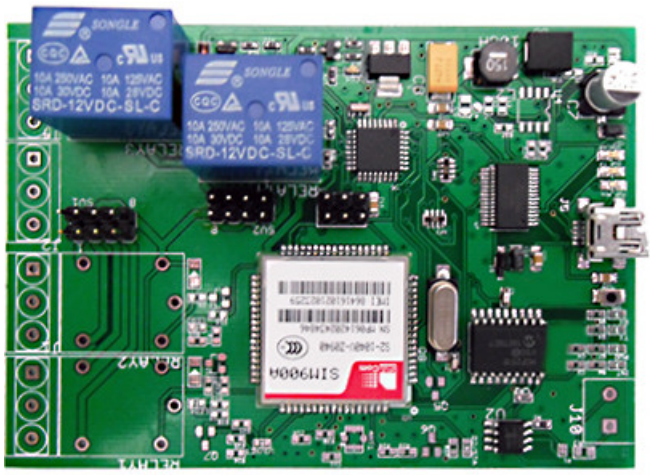


Figure 2: Arducc board layout without two relays

2.2 Tasks and Runtime Library in Ada

A considerable number of debates has been done amongst programmers, language designers and operating system designers about operating-systems-defined versus language-defined concurrency [15, p. 29]. The Ada language provides a built-in concurrent model that can be used to discourage some unpredictable behaviors through task types, protected types, rendezvous, selective wait, and guards. Although the concurrent model is designed in Ada as tasks, these are commonly implemented in the runtime library as thread primitives of some operating system. Threads are commonly associated to concurrent entities of operating systems that can run sequentially or in parallel. According to Burns and Wellings [15] the term coroutine appears more appropriate to language-defined concurrency based on uni-processor systems (as the case of this work that uses 8-bit AVR MCUs with one core [2]), tasks to language-defined concurrency supporting multi-processor systems, and threads to operating systems processes.

2.2.1 Task Types

Task types are expansions that are treated by the GNAT compiler and converted into a set of Ada primitives without task types [16]. Ada has a proper syntax for task declaration as shown in Listing 1, where AVR_Single is the name of the declared task. The initialization of a task without a discriminant is the same as declaring a variable, 'T: AVR_Single;'. Although Ada does not have syntax for multiple entries and multiple returns, the syntax is elegant and adequate to define and manage coroutines in AVR micro-controllers without excessive overhead and memory footprint. Coroutines are primitives useful for predictable context-switching programs.

2.2.2 Candidates for Task Types Primitives

The AVR Thread Library [17] is mainly designed in a mix of C and assembly. The interrupt service routines (ISRs) are implemented in AVR assembly language, and the remaining functions are fully implemented in C. The interface between both programs is given at linker level as well. avr_thread_init is a function to initialize the structures for supporting task

```
task type AVR_Single is
   -- <Declarations> of exported identifiers
end AVR_Single;

task body AVR_Single is
   -- local <Declarations>
begin
   -- <Task Statements>
end AVR_Single;
```

Listing 1: Template for a single task

context-switches, allowing the addition or removal of new tasks to the context switch structure by the avr_thread_start or avr_thread_stop functions. Changed the context-switch structure, any task can be forced to context-switch with avr_thread_sleep or avr_thread_yield. Each task is scheduled by a Round Robin policy, and the scheduler is executed when a tick is triggered by an hardware timer with a certain period setting. avr_thread_isr_start and avr_thread_isr_end are ISRs that will be used to execute the scheduler when the hardware timer triggers. Thread synchronization and signaling is supported by this library as well. Mutexes are mutual exclusion semaphores to guarantee that only one thread is executing a particular piece of code at a time, and events provide the way to signal to other threads that an event has occurred.

To support hard real-time tasks this thread library is not enough. Chibios/RT RTOS [18] is an alternative to that process but suffers significant overheads and memory footprint mainly due to the heavy data structures required by the abstraction library (HAL). Our criteria is to take advantage of both worlds and extend Ada tasks supporting timed constraints (e.g., deadline and period) as well as temporal constraints to ensure some type of temporal order. The hard real-time scheduling uses a fixed-priority scheduler policy.

2.2.3 The AVR-Ada Runtime System

AVR-Ada is composed by an AVR support library, an Ada runtime library, and several patches for the GCC based Ada compiler (GNAT). The support library contains drivers for external peripherals such as Dallas' 1-Wire sensors and components, Sensirion temperature and humidity sensors, LCDs based on HD44780 controllers, SPI support for the MCP4922 DAC chip, and MIDI support for any byte stream. Other pieces of code enable supporting FAT16/32 filesystems for SD/MMC cards, the generation of CRC8 and CRC16 values, and the support for the slip protocol that packets I/O over streams as asynchronous serial links. The Ada runtime library supports several ATMega devices such as the *ATMega328p* that is the MCU used by the Arducc board and by the Arduino boards and derivatives. GNAT is officially distributed as a standalone package by the AdaCore company providing a full compiler for AVR platforms with a zero-footprint runtime library, and is also partially included in the official release of GCC. A complement of such limiting profile is surpassed by AVR-Ada patches that enables runtime features for AVR 8-bit micro-controllers. Both AVR-Ada and AdaCore AVR compilers do not support multi-task scheduling neither real-time features such as timing events and execution time timers.

```
GNARL.Enter_Master;
task type avr_single;
avr_singleE : aliased boolean := false;
avr_singleZ: size_type := GNARL.Unspecified_Size;

type avr_singleV is limited record
   _task_id : task_id;
end record;

procedure avr_singleB (_Task: access avr_singleV);

freeze avr_singleV [
   procedure avr_singleVIP ( ... ) is
   begin
      GNARL.Create_Task ( ... );
   end avr_singleVIP;
]

procedure avr_singleB (_task: access avr_singleV) is
  procedure _Clean is
  begin
    GNARL.Abort_Defer;
    GNARL.Complete_Task;
    GNARL.Abort_Undefer;
  end _Clean;
  begin
    GNARL.Abort_Undefer;
    -- <Declarations>
    GNATRL.Complete_Activation;
    -- <Task Statements>
    _Clean;
end AVR_SingleB;
```

Listing 2: Expansion of Task Type

3 Implementation

3.1 Expansion of Ada Task Types

Tasks are expanded by the compiler into a set of procedures that have been deployed in the Ada runtime library. This feature allows Ada to support a native concurrent model for different architectures. For AVR, redesigning the task primitives is essential to overcome the memory footprint.

The compiler for Ada expands task types without a discriminant into a set of primitive types and some procedures [16], as shown in Listing 2. GNARL.Create_Task is the procedure that adds the task into the available structures; _Clean is a procedure to instruct the scheduler that a task has been finished, should be removed from support data structures, and does not still awake; GNARL.Abort_Defer and GNARL.Abort_Undefer instruct the deferral state; and GNARL.Complete_Activation consolidate the activation of a task, indicating that it is now ready to run.

3.1.1 Dependencies

For our proposal of supporting tasks in AVR-Ada, we will describe for each package its purpose. The packages that we need to consider adding or modifying for AVR-Ada are, as follows:

- System.Tasking.Initialization is a package that is required for Init_RTS procedure, and to initialize the Ada records required for task context-switches.
- System.Tasking provides type definitions for compiler interface such as Task_States, Activation_Chain, and Ada_Task_Control_Block.
- System.Tasking.Stages contains the procedures that the compiler introduces after each task type extension, such as Create_Task, Activate_Tasks, Complete_Activation, and Complete_Task.
- System.Secondary_Stack contains the procedures to manage the secondary stack which is used when tasks are allowed. It is capable to mark the stack with symbols for management of sub-stacks.
- Ada.Real_Time is the most interesting package for real-time programming. It is not only used by runtime library but aids the developer in using the operation *delay until* within tasks.
- Ada.Real_Time.Timing_Events is another interesting package that implements the software timers based on a main periodic task.
- System.Soft_Links is a package containing several stack pointer definitions. It contains also a set of subprogram access variables that calls different primitives depending whether tasking is involved or not.
- System.Stack_Checking is a package very useful in embedded systems for counting the usage of the stack. It is system-independent.
- System.Task_Info is used as a dummy package because GNAT compiler requires such interface. It includes definitions and procedures for primitives that concurrent model will use in the operating system.
- System.Multiprocessors is another dummy package. It is required for multi-core systems but we only use it to surpass the compiler dependencies of the GNARL.Create_Task procedure arguments.

3.2 Coroutines and Scheduler

Coroutines are a generalization of subroutines containing multiple entry points and multiple return points that can progress at same time but not on the same time instant. Basically, they are a form of concurrent processing that is used in several programming languages such as Python [19] and Lua [20] to establish cooperative multitasking [21].

The scheduler for coroutines allows managing the execution policy over a set of coroutines. It can be preemptive for real-time applications containing a priority level system, or non preemptive in the form of cooperative scheduling. The main advantage of preemptive scheduling is the real-time response on the task level, and with cooperative scheduling substantially fewer uncontrollable switches are encountered than in preemptive scheduling, because tasks cannot be preempted.

To subdivide the functionalities, we manage these terms into two new packages:

- System.Coroutine provides the yield and resume of any coroutine entity; and
- System.Scheduler provides the scheduler policies definition, the update state procedure, and several support types and procedures for scheduling.

```
procedure Context_Switch is
begin
   Store_Context;
   System.Scheduler.Update_State;
   Restore_Context;
end Context_Switch;
```

Listing 3: Task Context-Switch Procedure

```
procedure Store_Context is
begin
   -- push into stack r1-r31 registers
   Asm ("push r1" & ASCII.LF & (....)  "push r31",
           Volatile  => True);
   -- push into stack SREG with global interrupts enabled
   Asm ("in r26, __SREG__" & ASCII.LF &
        "sbr r26, 128" & ASCII.LF &
        "push r26",
           Volatile  => True);
   -- save current Stack Pointer into variable Stack_Pointer
   Asm ("in r26, __SP_L__" & ASCII.LF &
        "in r27, __SP_H__" & ASCII.LF &
        "std Y+0, r26" & ASCII.LF &
        "std Y+1, r27",
           Inputs => System.Address'Asm_Input (
                        "y", Stack_Pointer'Address),
           Volatile  => True);
   -- push SREG to be used after Update_State call [X]
   Asm ("in __tmp_reg__, __SREG__" & ASCII.LF &
        "push __tmp_reg__",
           Volatile  => True);
end Store_Context;
```

Listing 4: Procedure to store the context of a task

3.2.1 *Context-switch*

The context-switch procedure that is used to change context from a set of coroutines is provided by the package System.Coroutine and has the body as shown in Listing 3. System.Scheduler.Update_State is an inline function to update the state of the scheduler such as the tick variable, and the other procedures store and restore the context. In addition, the Context_Switch is implemented as a naked procedure using the Machine_Code pragma that instruct the compiler to remove any extra return code or context for registers.

The store procedure of the context is defined using a mix of Ada and assembly code as shown in Listing 4. Unsigned_8'Asm_Input is the type of the input, the string "z" establishes that the variable of the type Unsigned_8 is assigned to the Z register [22], and the function Current_Coroutine returns the address of the memory space to save the current stack pointer. In sum, this block pushes into the stack all MCU registers (r1 to r31) and the status register (SREG) with global interrupts flag activated, and save the current address of the stack pointer into the context structure.

The restore of the context for the new coroutine returned by the function New_Coroutine is defined by a set of assembly statements in Ada as shown in Listing 5. New_Coroutine function return the address of the new stack pointer. In short, this block restores the new task to execute assigning the registers that have been saved as the current registers as well as the SREG.

```
procedure Restore_Context is
begin
   -- restore SREG from previous push [X]
   Asm ("pop __tmp_reg__" & ASCII.LF &
        "out __SREG__, __tmp_reg__",
           Volatile  => True);
   -- set the new stack pointer and status register
   Asm ("ldd r26, Y+0" & ASCII.LF &
        "ldd r27, Y+1" & ASCII.LF &
        "out __SP_L__, r26" & ASCII.LF &
        "out __SP_H__, r27" & ASCII.LF &
        "pop __tmp_reg__" & ASCII.LF &
        "out __SREG__, __tmp_reg__",
           Inputs => System.Address'Asm_Input (
                        "y", Stack_Pointer'Address),
           Volatile  => True);
   -- push into stack r1-r31 registers
   Asm ("pop r31" & ASCII.LF & (...)  "pop r1",
           Volatile  => True);
   -- return and set global interrupts on
   Asm ("reti",
           Volatile  => True);
end Restore_Context;
```

Listing 5: Procedure to restore the context of a task

3.2.2 *Scheduler Policies*

Fixed-priority scheduling is considered by the real-time community more flexible than cyclic scheduling but more restricted than dynamic scheduling [23, 24, 25]. This algorithm ensure that at any given time instant the processor is executing the highest priority task. To do that the scheduler is triggered at a certain period, which is normally named as time slice or quantum, by a timer interrupt routine. The interrupt is used to run the scheduler periodically, and to execute a context-switch from the current task to the new task when applicable. Normally, the scheduler uses two queues: one ready queue and one waiting queue. The ready queue contains the tasks that are ready for execution while the waiting queue contains the tasks that have been executed and now are awaiting their execution. In short, the execution rules that are considered at each time slice are:

- if a task from the waiting queue becomes ready for execution, then it is moved to the ready queue according to its priority;
- if a task is running and one waiting action is triggered the task is inserted in the waiting queue ordered by the remaining time to execute;
- if the priority of the first task from the ready queue becomes higher than the priority of the current task, then a context switch is done.

Choosing the time slice parameter is critical for balancing the system performance and the task responsiveness – choosing a very short value can compromise the processing power of the system, and a very huge value can overpass the deadlines of periodic tasks.

This policy is implemented as an inline procedure that change the state of the scheduler to be activated, and is defined in the package System.Scheduler as shown in Listing 6. This procedure assumes that the ready_queue is ordered by priority.

Round-Robin scheduling is considered a form of cooperative task model. It is well used in non real-time applications to fairly share execution time between tasks. The rules of this type of scheduler are simple. Basically, at each time slice the system instead of triggering the higher priority task, only triggers the next task to execute. The tasks form a chain that is switched at each schedule tick. The order can be clockwise or counterclockwise.

This policy is implemented as an inline procedure that change the state of the scheduler to activate a context-switch, and is defined in the package System.Scheduler as shown in Listing 7.

3.2.3 Interrupt Routine

Embedded systems make an intensive use of interrupts and even more of the ones generated by hardware timers. AVR architectures have few timers and then depending on the application they need to be carefully managed. This limitation can be bypassed by the tick method. It manages an enormous amount of timers as required by real-time systems and supports the generation of software timers that are charged to call a specified handler when they finish. The overhead is not significant, and the precision is as good as the one given for each tick. Tasks are scheduled using these software timers, one for each task, and they are named timing events when coupled with a handler.

For safety purposes, the nested interrupts need to be taking into account and almost always predictable as well as the conditions for context-switching. The author describes the rules that can result in a change of context from a scheduler, as follows:

- if the task is executing and some procedure is called to suspend it until a certain amount of time, a new timing event is created, the handler attached and triggered when the timing event expires a certain amount of time, and the task removed from the ready queue;
- if the task is executing and any procedure is called to suspend the task within an undetermined amount of time then no timing event triggers, and the task may not wake-up anymore if the programmer does not insert any method to do it explicitly;
- if no suspensions applied, only the handler can context-switch tasks.

```
procedure FP_Policy is
begin
   -- compare current task priority with the first task in
   -- priority queue
   if ready_queue.priority < current_task. priority
      -- do nothing
      State := True;
   else
      -- do the context switch
      State := False;
   end if;

end FP_Policy;
```

Listing 6: Procedure to select the next task to execute using the fixed-priority scheduling policy

```
procedure RR_Policy is
begin
   -- always do the context switch
   State := False;
end RR_Policy;
```

Listing 7: Procedure to select the next task to execute using the Round-Robin scheduling policy

```
package Ada.Real_Time.Timing_Events is
   type Timing_Event is tagged limited private;
   type Timing_Event_Handler is access
     procedure (Event : in out Timing_Event);
   procedure Set_Handler (Event : in out Timing_Event;
                          At_Time : in Time;
                          Handler : in Timing_Event_Handler);
   procedure Cancel_Handler (Event : in out Timing_Event;
                             Cancelled : out Boolean);
   -- the new procedure to be called only from Runtime Library
   procedure Do_Tick_and_Execute;
private
   -- define the timing event record
   type Timing_Event is limited record
      Timeout : Time := Time_First;
      Handler : Timing_Event_Handler;
      Handler_Arg : System.Address;
      Next : access Timing_Event := null;
      Previous : access Timing_Event := null;
   end record;
   -- define a list of timing events for Do_Tick_and_Execute
   Timing_Events_list : access Timing_Event;
   Time : Time := 0;
end Ada.Real_Time.Timing_Events;
```

Listing 8: Timing Events

The implementation of timing events is part of the package Ada.Real_Time.Timing_Events, and the specification is as shown in the Listing 8. At scheduler level, the runtime library uses the timing events that are responsible for coordinating real-time tasks.

The scheduler handler is part of the package System.Scheduler.Timer_Handler, and a hardware timer provides the interrupt routine to call it. Listing 9 shows the code to define the settings for this interruption routine, where X is the identifier of the interrupt vector that can be a number from 0 to 16 depending on the MCU, and __vector_X is the interrupt vector to link with the handler. The tick respects some precision, and the scheduler executes in this tick granularity configured by one of the available timers, which are hardware dependent.

Timer_Handler routine is as described in the Listing 10. It executes an entirely isolated procedure, which means that every context of the used registers are saved and replaced after the execution, and takes a decision if any task can change and when positive make the context-switch. For now, the Isolated_Execution procedure only contains the call to the procedure Do_Tick_and_Execute for timing events and the procedure Policy. Do_Tick_and_Execute routine is capable to update every event timer and execute their handles if remaining time have been expired as shown in the Listing 11. The Handler is the procedure that will put available the task into the ready queue.

```
procedure Timer_Handler;
pragma Machine_Attribute (Entity => Timer_Handler,
                          Attribute_Name => "signal");
pragma Export (C, Timer_Handler, "__vector_X");
```

Listing 9: Interrupt handler definition

```
procedure Timer_Handler is
   use System.Machine_Code;
begin
   -- save and restore the registers used by it
   Isolated_Execution;
   -- disable interrupts enabled in the "reti" of the last
   -- procedure
   Asm ("cli", Volatile => True);
  -- make a decision to do the context switch
   Asm ("push r26" & ASCII.LF &
        "ld r26, Y" & ASCII.LF &
        "tst r26" & ASCII.LF &
        "brne .nn" & ASCII.LF &
        "pop r26" & ASCII.LF &
        "reti" & ASCII.LF &
        ".nn:" & ASCII.LF &
        "pop r26",
           Inputs => System.Address'Asm_Input (
                        "y", State'Address),
            Volatile => True);

   -- do the context-switch
   Context_Switch;
   -- part that is no longer achievable
end Timer_Handler;
```

Listing 10: Timer handler

3.3 Interconnection with AVR-Ada

3.3.1 Required Data Types

Data structures to save the context and the state of a task are defined using the guidelines provided in the GNAT book [16]. However, to reduce memory footprint and avoid safety problems, all definitions for entries, activations, and dynamic allocators have been disabled.

Tasks are coroutines containing a state variable to inform the scheduler and a stack for saving the context of the tasks including the outcomes from the push and pop instructions made in the task body. Listing 12 describes these states. The state Ready means that the task is waiting in the ready queue; the state Runnable means that the task is currently executing; the state Terminated means that the task cannot have more executions, and the state Sleeping establishes that the task is waiting from a delay statement. The stack pointer is saved in the control block of the task as well as other fields such as, the priority of the task for fixed-priority policies and a pointer to the procedure expanded by the compiler from the task type. This control block is denoted by Ada_Task_Control_Block as shown in the Listing 13. State contains one of the available task states; Parent contains the address of the task that elaborated the new task; Base_Priority contains the priority of the task; Current_Priority contains the priority for the case of ceiling locks can happen; Task_Arg contains an address for the task arguments; Task_Entry_Point contains the address of the task procedure, and the remaining attributes are to match the links and interfaces provided by the GNAT compiler.

```
procedure Do_Tick_and_Execute is
   Tmp : access Timing_Event := null;
begin
   -- increment the time
   Time := Time + 1;
   if not Timing_Events_list = null
      -- decrements all timers
      Timing_Events_list.Timeout := Timing_Events_list.Timeout - 1;
      while not Timing_Events_list.Next = null loop
         Tmp := Timing_Events_list.Next;
         Tmp.Timeout := Tmp.Timeout - 1;
      end loop;
      -- is zero
      if Timing_Events_list.Timeout = 0
         -- execute handler
         Timing_Events_list.Handler;
      end if;
   end if;

end Do_Tick_and_Execute;
```

Listing 11: Do_Tick_and_Execute **procedure**

```
type Task_States is ( Ready, Runnable, Terminated, Sleeping );
```

Listing 12: Task states for AVR-Ada

3.3.2 Secondary Stack

The local stack for each task is assigned to a segment of the secondary stack and is separated in memory by proper symbols. The allocated space is not available anymore. This constraint prevents the use of pooling methods for dynamic memory allocation, reduces the significant memory footprints for AVR architectures with tiny SDRAM, and increases the predictability and the safety of the embedded systems. As already referenced in the GNAT runtime library documentation [16], Task_Id is an access type to store the address of the task control block. Usually, the control block is created using dynamic memory allocation and the static, limited record provided by task expansion stores the address. In this work, the task type contains the initial address provided by the secondary stack to store the control block. The new stack pointer increases with the size of this block making the initial stack segment used and avoiding task block overwrite. The statement 'for Control_Block'Address use Local_Stack'Address;' modifies the address of the control block, and the stack pointer need to be shifted using an assembler instruction.

```
type Ada_Task_Control_Block is record
   State : Task_States;
   Parent : Task_ID;
   Base_Priority : System.Any_Priority;
   -- priority ceiling
   Current_Priority : System.Any_Priority;
   -- task arguments and task procedure access
   Task_Arg : System.Address;
   Task_Entry_Point : Task_Procedure_Access;
   -- structure to store stack addresses
   Compiler_Data : System.Soft_Links.TSD;
   All_Tasks_Link : Task_ID;
   Elaborated : Access_Boolean;
end record;
```

Listing 13: AVR-Ada task control block

This remark intended to discourage programmers to use tasks in AVR for running a dozen times due to the significant stack footprint. It gives a burden for the programmer to chose how to deal with aperiodic tasks.

3.3.3 Tasking Stages

Activate_Tasks and Complete_Ativation are declared as inline procedures that contains the instruction 'null;'. We do not need these functions since we do not use any operating system, and no synchronization need to be ensured between the elaboration and the first execution of a task.

RTS_Lock and RTS_Unlock are procedures to activate or deactivate tasks. As we know only an interrupt can context-switch the current task to another task, and then we disable it for a while. However, we need to take care about the usage of these functions.

Create_Task is the procedure that calls another function defined in the package System.Coroutines. A task is mapped to a coroutine that have its proper stack allocated as a sub-stack of the secondary stack. Terminate_Task is a null procedure since we consider that any task does not ends due to the secondary stack release constraint.

```
type Controller_State_Type is (Sleep, Halt, Waiting_For_Halt,
  Delay_By_Flag, Auto_Start_Up, Auto_Shutdown, Force_Shutdown);
for Controller_State_Type use (Sleep => 16#01#, Halt => 16#02#,
  Waiting_For_Halt => 16#04#, Delay_By_Flag => 16#08#,
  Auto_Start_Up => 16#10#, Auto_Shutdown => 16#20#,
  Force_Shutdown => 16#40#);

type Relay_Block_State_Type is (Computer, Computer_Ready,
  Monitor, Amplifier );
for Controller_State_Type use (Computer => 16#01#,
  Computer_Ready => 16#02#, Monitor => 16#04#,
  Amplifier => 16#08#);

type Can_Block_State_Type is (Can_Monitor);
for Can_Block_State_Type use (Can_Monitor => 16#01#);

type Vehicle_Block_State_Type is (Lights, Reverse_Gear, Engine,
  Driver_Door, Ignition );
for Vehicle_Block_State_Type use (Lights => 16#01#,
  Reverse_Gear => 16#02#, Engine => 16#04#,
  Driver_Door => 16#08#, Ignition => 16#10#);

type States is record
   Controller_State: Controller_State_Type;
   Relay_Block_State: Relay_Block_State_Type;
   Can_Block_State: Can_Block_State_Type;
   Vehicle_Block_State: Vehicle_Block_State_Type;
end record;
```

Listing 14: A preliminary state version for Arducc board.

4 A Lightweight Control System

Arducc board is a platform designed explicitly for lightweight automotive applications. The controller system that runs on this platform composed by three tasks manages different modules provided by the Arducc board such as CAN, GSM, and GPS interfaces. The control system is purely reactive and acts to state changes done by any task. The task that the paper presents in detail is the CAN task that uses a specific library to communicate via SPI bus with the MCP2515 transceiver.

4.1 Preliminary Definitions

The task type Can_Task is the CAN interface containing the initialization of the CAN driver and the parsing of the CAN messages that sets according to the CAN messages the respective state change of the system. GSM_Task is used to control the GSM module included in the Arducc board, and Controller_Task is the main controller task to turn-on/-off the monitor, computer, and other devices coupled to the Arducc board. The state change modifies the state vector of size 4 bytes, as shown in the Listing 14. It initializes a predefined setting saved in the EEPROM of the MCU. To ease the next analysis, ϵ_1 denotes the Can_Task, and ϵ_2 the Controller_Task.

4.2 CAN Task

The messages produced by the vehicle and parsed in the Can_Task uses the MCP2515 external library, as shown in the Listing 15. This periodic task does the parsing by reading a message buffered in an FIFO list provided by MCP2515 library. The library supports the poll method for high rating message monitoring and interrupt method for low rate messages. In interrupt mode, the messages are filtered with a certain mask id by the procedure Mcp2515.Set_Mask, and the library statically allocates a buffer of ten messages to be consumed by tasks. The standard CAN messages have an identification of 11bits, a maximum of 8 bytes for the message, a CRC check code, and some acknowledgment bits [26]. The minimum size of the standard CAN messages is 44bits and can be extended to 108bits to carry the message of 8 bytes.

The task begins by initializing the MCP2515 driver, sets filters for two hexadecimal addresses 0x380 and 0x3A0, requests the state of the FIFO list, and consumes it whether a list is not empty. The period assigned for this consumption process needs an optimization to avoid missing any message based on the maximum message arrival rate. The case statement matches the acquired message identification with a certain state according to the message content. The mutual exclusion approach used to ensure that any interrupt does not interfere with getting a message requires taking into account the deactivation of the interrupts, since the interrupts are unable for a long time, a miss of the tick of the scheduler can occur. The common and safe method is applying a mutex to a certain variable, but currently this feature is not supported by AVR-Ada. The other two tasks are not presented here but can be found in the next release of Arducc firmware [14].

4.3 Interrupt Handling

The schedulability of the tasks depends on an interrupt that is triggered by a hardware timer. As the hardware timers are dependent of the MCU, an extra setup for each MCU provided by Ada AVR library implements the periodic call of the scheduler handler in the runtime library. This extension provides to AVR-Ada a configuration of the Timer1 in ATMega328 as shown in the Listing 16. The procedure Timer1_Setup setups the registers of the timer one to trigger an interrupt twenty times per second. However, this approach is not fully implemented to work with all AVR chips supported by AVR-Ada.

```
task body Can_Task is
   tmp_msg   : Mcp2515.Message_Type;
   Period    : constant Ada.Real_Time.Time_Span :=
     Ada.Real_Time.Milliseconds (40);
   Next_Time : Ada.Real_Time.Time := Ada.Real_Time.Clock;
begin
    Mcp2515.Init(Mcp2515.Baund_50kbs);
    Mcp2515.Set_Mask(16#380#);
    Mcp2515.Set_Mask(16#3A0#);
   loop
      Ada.Real_Time.Lock;
      if not Mcp2515.Rx_Buffer.Empty then
         tmp_msg := Mcp2515.Rx_Buffer.Get;
         Ada.Real_Time.Unlock;
         -- begin of CAN messages parser
         case tmp_msg.id is
            when 16#380# =>
               -- dashboard system message
               if tmp_msg.data (2) = 16#04# then
                  Marv.State.Vehicle_Block_State (Marv.
                    Vehicle_Block_State_Bit_Type.
                    Driver_Door) := True;
               end if;
            when 16#3A0# =>
               -- multimedia system message
               if tmp_msg.data (1) = 16#16# then
                  Marv.State.Multimedia_Block_State (Marv.
                    Multimedia_Block_State_Bit_Type.
                    Next_Music) := True;
               end if;
         end case;
         -- end of parser
      else
         -- release the lock
         Ada.Real_Time.Unlock;
      end if;
      -- periodic delay
      Next_Time := Next_Time + Period;
      delay until Next_Time;
   end loop;
end Can_Task;
```

Listing 15: Parsing CAN messages for Arducc board.

```
with AVR; use AVR;
with AVR.MCU;
procedure Timer1_Setup is
begin
   MCU.TCCR1A := MCU.TCCR1B := MCU.TCNT1 := 0;
   -- set the 256 prescaler
   MCU.TCCR1B_Bits (CS12_Bit) := True;
   -- 16MHz(clock)/256(prescaler)/20Hz(interrupt frequency)
   -- (16_000_000/256)/20 = 3125
   MCU.OCR1A := 3125;
   MCU.TCCR1B_Bits (WGM12_Bit) := True; -- CTC mode
   -- enable Timer1 compare interrupt
   MCU.TIMSK1_Bits (OCIE1A_Bit) := True;
end Timer1_Setup;
```

Listing 16: Extra code for ATMega328 Timer1 setup.

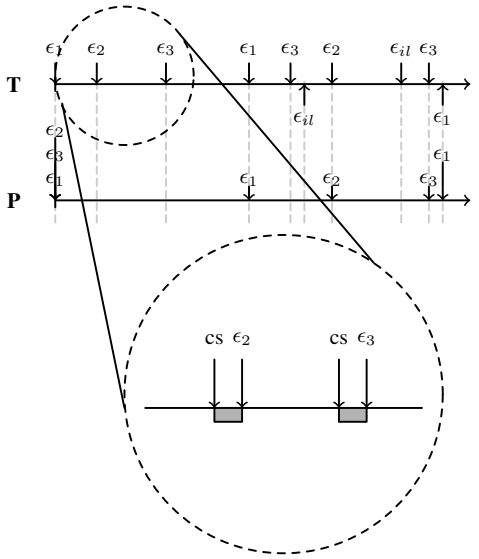


Figure 3: Illustration of the context-switches generated from a set of tasks using a fixed-priority scheduler.

For users that do not use ATMega MCUs, a few lines of code need to be added to activate one timer to trigger the Time_Handler at certain period.

4.4 Concurrency Analysis

Consider a set of tasks $\{\epsilon_{il}, \epsilon_1, \epsilon_2, \epsilon_3\}$ with implicit deadline containing the period value of $d_{\epsilon_1} = 14$, $d_{\epsilon_2} = 20$, and $d_{\epsilon_3} = 27$. For a scheduler using a fixed priority policy the priorities are assigned with the following order $p_{\epsilon_{il}} < p_{\epsilon_3} < p_{\epsilon_2} < p_{\epsilon_1}$, as shown by the time sequence in the Figure 3. P is the pattern where a task should be released, and T the trace showing the sequential execution for some time instants. ϵ_{il} describes that any task is not executing, and the system is in idle mode. This task uses a very small stack allocated in the beginning of the secondary stack, and the other tasks have sub-stacks allocated in the secondary stack, as shown in the Figure 4. The arrows identifies the four possible addresses that can be assigned to the stack pointer in SDRAM, the SDRAM on top shows the memory assignment without the elaboration of the tasks, and the bottom SDRAM piece depicts the memory assignment after the tasks elaboration. The .data and .bss sections are defined as usual.

Focusing on the first three events of the trace, the reader can realize that before each event a context switch is done by the scheduler. This context-switch copy every register of the MCU into the current stack pointer, and the current stack pointer into the task control block. The behavior of the system with these task settings is, as follows:

1. any task has an access of task control block as the task identifier;
2. every task is initialized with 31 registers pushed into stack with value 0 and SREG with the default value;
3. the current local stack pointer address is saved in the task control block for each task;
4. the task ϵ_1 starts the execution by context-switching the registers and the stack pointer assigned in the initialization process;
5. the context from ϵ_1 into ϵ_2 pushes the current registers r1-r31 and SREG into the local stack $stack(\epsilon_1)$, saves the current stack pointer into the task control block, updates the scheduler state increasing the timers, and does the restore context by popping all registers from local stack of the task ϵ_2;

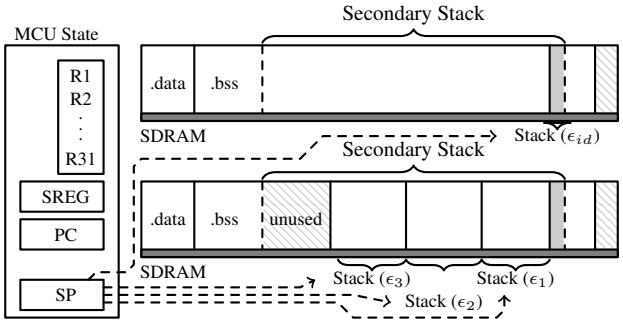


Figure 4: A picture of MCU and memory assignment including four stacks.

6. the process continues with task ϵ_2 into the step 4), where at each cycle the task identifier is changing.
7. when no task have to be processed the system enters in idle mode by continuously executing the idle task ϵ_{il}, and when a task is ready continues to step 4).

The scheduler tick granularity may have tried several times to perform some context switches during the execution of the system, but the timer handler does not instruct the scheduler to do it due to the priority of the task under execution.

5 Conclusion

This paper presented an extension of AVR-Ada for AVR 8-bit microcontrollers, with support for task types, timing events, and simple lock/unlock features. Accepted scheduling policies are Round-Robin and Fixed-Priority, but other policies can be added easily to the runtime library of AVR-Ada by changing the policy procedure. This work allows programmers to design efficient real-time Ada applications for MCUs with subtle memory RAM as the case of AVR 8-bit microcontrollers. The major advantages of this extension are the design of lightweight embedded systems with very low consumption MCUs using a concurrent language, a small and static memory usage due to optimizations done inlining assembly code, and the support for verification of embedded systems using Ada 2012 contracts.

For future work, we address the extension of protected types as concurrent monitors, the design of execution time timers, and the introduction of the Ada 2012 contracts for embedded systems design phase. Contracts should be checked dynamically extending the runtime system and the compiler.

References

[1] Atmel® AVR® 8-bit/32-bit microcontrollers Family. http://www.atmel.com/products/microcontrollers/avr/ Accessed: 5/08/2014.

[2] Atmel® AVR® 8-bit microcontrollers Overview. http://www.atmel.com/images/45058a_about-avr_090913.pdf Accessed: 5/08/2014.

[3] Arduino: An opensource board based on Atmel® AVR® 8-bit microcontrollers. http://www.arduino.cc/ Accessed: 5/08/2014.

[4] Ardupilot: An opensource autopilot based on Arduino project. http://www.ardupilot.com/ Accessed: 5/08/2014.

[5] Ardusat: The first opensource satellite based on a mesh of Atmel® AVR® 8-bit microcontrollers. http://www.ardusat.org/ Accessed: 5/08/2014.

[6] K. N. Gregertsen and A. Skavhaug (2010), *Implementing the New Ada 2005 Timing Event and Execution Time Control Features on the AVR32 Architecture*, J. Syst. Archit., vol. 56, pp. 509–522.

[7] P. V. Rego (2012), *Integrating 8-bit AVR Micro-Controllers in Ada*, Ada User Journal, vol. 33, pp. 301–305.

[8] AVR-Ada: The Runtime Library for Atmel® AVR® 8-bit microcontrollers. http://sourceforge.net/projects/avr-ada/ Accessed: 5/08/2014.

[9] Ada Development Environment for the Atmel® AVR® 8-bit microcontroller – AdaCore. http://www.adacore.com/gnatpro/embedded/avr/ Accessed: 5/08/2014.

[10] Eclipse: An integrated development environment. https://www.eclipse.org/ Accessed: 5/08/2014.

[11] GPS: The GNAT Programming Studio. http://libre.adacore.com/tools/gps/ Accessed: 5/08/2014.

[12] The AVR Eclipse Plugin. http://avr-eclipse.sourceforge.net/ Accessed: 5/08/2014.

[13] C. Marlin (1980), *Coroutines: A Programming Methodology, a Language Design and an Implementation*, Lecture Notes in Computer Science, Springer.

[14] Open Hardware Arducc Board Project. http://sourceforge.net/projects/arducc/ Accessed: 5/08/2014.

[15] A. Burns and A. Wellings (2007), *Concurrent and Real-Time Programming in Ada*. New York, NY, USA, Cambridge University Press, 3rev ed.

[16] AdaCore, GNAT: The GNU Ada Compiler Book. www.adacore.com/gap-static/GNAT_Book/gnat-book.pdf Accessed: 5/08/2014.

[17] AVR Threads Library. https://bitbucket.org/dferreyra/avr-threads Accessed: 5/08/2014.

[18] ChibiOS/RT: An open source, compact and extremely fast RTOS. http://www.chibios.org Accessed: 5/08/2014.

[19] M. Lutz (2003), *Learning Python*. Sebastopol, CA, USA, O'Reilly & Associates, Inc., 2 ed.

[20] R. Ierusalimschy (2013), *Programming in Lua, Third Edition*. Lua.Org, 3rd ed.

[21] A. Adya, J. Howell, M. Theimer, W. J. Bolosky, and J. R. Douceur (2002), *Cooperative Task Management Without Manual Stack Management*, Proceedings of the General Track of the Annual Conference on USENIX Annual Technical Conference, ATEC '02, (Berkeley, CA, USA), pp. 289–302, USENIX Association.

[22] Atmel® AVR® 8-bit RISC Instruction Set. `http://www.atmel.com/Images/Atmel-0856-AVR-Instruction-Set-Manual.pdf` Accessed: 5/08/2014.

[23] C. D. Locke (1992), *Software Architecture for Hard Real-time Applications: Cyclic Executives vs. Fixed Priority Executives*, Real-Time Syst., vol. 4, pp. 37–53.

[24] G. C. Buttazzo (2005), *Rate Monotonic vs. EDF: Judgment Day*, Real-Time Syst., vol. 29, pp. 5–26.

[25] P. A. Laplante and S. J. Ovaska (2011), *Real-Time Systems Design and Analysis: Tools for the Practitioner*. Wiley-IEEE Press, 4th ed.

[26] Stand-Alone CAN Controller with SPI Interface – Microchip. `http://ww1.microchip.com/downloads/en/DeviceDoc/21801d.pdf` Accessed: 5/08/2014.

SPARK 2014 Rationale: Data dependencies and Information Flow

Pavlos Efstathopoulos
Altran UK

Abstract

This paper continues the publication of the "SPARK 2014 Rationale", which started in the December 2013 issue of the Ada User Journal. In this instalment, we present two contributions regarding data dependencies and information flow in SPARK.

1 Data dependencies

Programs often use a few global variables. Global variables make passing common information between different parts of a program easier. By reading the specification of a subprogram we are able to see all of the parameters that the subprogram uses and, in Ada, we also get to know whether they are read, written or both. However, no information regarding the use of global variables is revealed by reading the specifications. In order to monitor and enforce which global variables a subprogram is allowed to use, SPARK 2014 has introduced the Global aspect, which I describe in this post.

I claim that aspect Global is extremely easy to use. In order to figure out whether I am right or wrong, lets conduct an experiment. Without having explained anything about how aspect Global works I will give you the specifications of three procedures and also the code of the Main that calls them. You will then have to guess if the Main will behave predictably or not. If you guess correctly, then my initial statement was correct (otherwise, ... oh well). So, here we go:

```
package Guess_The_Order is
   Global_Variable : Integer;

   procedure Print with
     Global => (Input  => Global_Variable);

   procedure Sub (X, Y : in Integer) with
     Global => (In_Out => Global_Variable);

   procedure Add (X, Y : in Integer) with
     Global => (Output => Global_Variable);

end Guess_The_Order;


with Guess_The_Order;
procedure Main is
begin
   Sub (6, 1);
   Add (2, 3);
   Print;
end Main;
```

The main program will produce unpredictable results because procedure Sub tries to read Global_Var before it has been initialized! The correct order would be, call Add first, Sub second and Print last. The Global aspect on procedure Add says that Global_Variable will be set by the subprogram. Procedure Sub's contract says that Global_Variable will be both read (which is what causes the problem) and written. Procedure Print only reads Global_Variable and presumably does some kind of printing.

Regardless of whether you guessed correctly or not, if you want to learn some more about how aspect Global works, read on!

The Global aspect helps us in two different ways. It ensures that:

- ALL global variables mentioned by the aspect, and ONLY them, will be used by the subprogram and
- global variables that are meant as inputs are not updated and global variables that are meant as outputs are NOT read before they are set.

The following four modes inform us of the way in which the subprogram interacts with its global variables:

- Input: A global variable of this mode can NOT have its value updated. The global variable can and HAS TO be read. This means that there has to be at least one execution path of the subprogram where the global is read. In essence, if we say that something is an Input, then it is an Input!
- Output: Global variables of this mode can NOT have their values read before the subprogram has set them. Furthermore, their values have to be set in ALL execution paths of the subprogram. This means that by the end of the subprogram the value of the global will always be set to something.
- In_Out: There has to be at least one execution path that reads the initial value of the global variable and one path that updates it.
- Proof_In: The value of the global variable can only be mentioned inside proof related annotations. Consequently, no outputs of the subprogram can ever be affected by a such a global variable. This mode allows us to use globals to check that certain properties are true for our subprogram but it also ensures that we did NOT accidentally affect the outputs of the subprogram while doing our checks.

When a mode is not specified, the tools assume a default mode of "Input". Conceptually, the first 3 modes are very similar to modes "in", "out" and "in out" of formal parameters.

Enforcing that NO global variables are used

When a subprogram is annotated with a "Global => null" contract, the tools will ensure that NO global variables are used by the subprogram. Be careful, having no Global contract is not the same as having a null Global contract (i.e. Global => null). In the absence of a Global contract, the tools will generate an implicit Global contract based on the subprogram's body, they will assume that to be the correct contract and will propagate it to all callers of the subprogram.

Correct Examples

```
package Correct_Globals with SPARK_Mode
is
  Everything_OK : Boolean := True;
  Global_Var : Natural := 1;
  Backup_Global_Var : Natural := 1;

  procedure Increase_1 (X : in out Natural) with
    Global => (Input => Global_Var);
  -- Same as: "with Global => Global_Var;"

  procedure Increase_2 (X : in out Natural) with
    Global => null;
  -- Cannot use any global variables here.

  procedure Increase_Global_Var (X : Natural) with
    Global => (In_Out   => Global_Var,
               Output   => Backup_Global_Var,
               Proof_In => Everything_OK);
  -- Everything_OK can only appear in
  -- proof-related annotations.
end Correct_Globals;

package body Correct_Globals with SPARK_Mode
is
  procedure Increase_1 (X : in out Natural) is
  begin
    X := X + Global_Var;
  end Increase_1;

  procedure Increase_2 (X : in out Natural) is
  begin
    X := X + 1;
  end Increase_2;

  procedure Increase_Global_Var (X : Natural) is
  begin
    pragma Assert (Everything_OK);
     -- Some assertion...
    Backup_Global_Var := Global_Var;
    Global_Var := Global_Var + X;
  end Increase_Global_Var;
end Correct_Globals;
```

Incorrect Examples

```
package Incorrect_Globals
  with SPARK_Mode
is
  Global_Var : Integer;

  procedure Read_Global_Variable
      (X : out Integer) with
    Global => Global_Var;

  procedure Update_Global_Var
      (X : Integer) with
    Global => (Output => Global_Var);

  function Increase (X : Integer) return Integer with
    Global => null;
end Incorrect_Globals;

package body Incorrect_Globals
  with SPARK_Mode
is
  procedure Read_Global_Variable (X : out Integer) is
  begin
    X := Global_Var;
    Global_Var := Global_Var + 1;
    -- Global_Var is of mode "Input"
    -- The error message generated by the tools is:
    -- "Global_Var" must be a global output of
    -- "Read_Global_Variable"
  end Read_Global_Variable;

  procedure Update_Global_Var (X : Integer) is
  begin
    if X < 0 then
      Global_Var := -1 * X;
    elsif X > 0 then
      Global_Var := X;
    end if;
    -- When X = 0, Global_Var is not being set. So it's
    -- Global mode should have been In_Out. Another
    -- way to look at it is as follows.
    -- A mode of "Output" means that the global variable
    -- is ALWAYS set while a mode of "In_Out" means
    -- that the variable is set  SOMETIMES BUT NOT
    -- ALWAYS. The warnigs printed by the tools here is:
    --   warning: "Global_Var" might not be initialized
  end Update_Global_Var;


  function Increase (X : Integer) return Integer is
    (if X < 0
     then X + 1
     else X + Global_Var);
  -- The error message generated by the tools is:
  -- "Global_Var" must be listed in the Global aspect
  -- of "Increase"
end Incorrect_Globals;
```

Generated Globals

When a user-provided contract is available, the tools will attempt to verify its validity and report back in case of a contradiction. As said before, in the absence of a user-provided contract, the tools generate a contract based on the body of the subprogram, they then consider this to be the "correct" contract and use it to verify any user-provided contracts.

```
package Generated_Globals with SPARK_Mode
is
  Global_Var : Integer := 1;

  procedure Without_Contract (X : out Integer);
  -- Based on the body, the tools will compute a global
  -- contract that will say that Global_Var is a global
  -- input

  procedure With_Correct_Contract
      (X : out Integer) with
    Global => Global_Var;

  procedure With_Incorrect_Contract
      (X : out Integer) with
    Global => null;
end Generated_Globals;

package body Generated_Globals with SPARK_Mode
is
  procedure Without_Contract (X : out Integer) is
  begin
    X := Global_Var;
  end Without_Contract;

  procedure With_Correct_Contract (X : out Integer) is
  begin
    Without_Contract (X);
    -- The computed Global contract verifies the validity
    -- of the user-provided contract of procedure
    -- With_Contract.
  end With_Correct_Contract;

  procedure With_Incorrect_Contract (X : out Integer) is
  begin
    Without_Contract (X);
    -- The computed Global contract contradicts the
    -- user-provided contract  of procedure
    -- With_Contract. Due to the mismatch, the following
    -- error is generated: "Global_Var" must be listed in
    -- the Global aspect of "With_Incorrect_Contract"
  end With_Incorrect_Contract;
end Generated_Globals;
```

Conclusion

In his book "Better Embedded Systems SW" Phil Koopman says: "The main problem with using global variables is that they create implicit couplings among various pieces of the program (various routines might set or modify a variable, while several more routines might read it). Those couplings are not well represented in the software design, and are not explicitly represented in the implementation language. This type of opaque data coupling among modules results in difficult to find and hard to understand bugs." We don't have this in SPARK, thanks to the Global aspect!

2 Information Flow

In a previous blog post we described how aspect Global can be used to designate the specific global variables that a subprogram has to read and write. So, by reading the specification of a subprogram that has been annotated with aspect Global we can see exactly which variables, both local and global, are read and/or written each time the subprogram is called. Based purely on the Global aspect, this pretty much summarizes the full extent of our knowledge about the flow of information in a subprogram. To be more precise, at this point, we know NOTHING about the interplay between the inputs and outputs of the subprogram. For all we know, all outputs could be randomly generated and the inputs might not contribute in the calculation of any of the outputs. To improve this situation, SPARK 2014 uses aspect Depends to capture the dependencies between a subprogram's outputs and inputs. This blog post demonstrates through some examples how aspect Depends can be used to facilitate correct flow of information through a subprogram.

We will start off with a simple example. Lets assume that we want to write a procedure that doubles and then swaps variables X and Y. The final value of X should depend only on the original value of Y and the final value of Y should depend only on the original value of X. So now let's write some code and add the depends contract that we just mentioned on it.

```
procedure Double_And_SWAP (X, Y : in out Integer)
  with Global  => null,  -- We use no global variables.
       Depends => (X => Y,
                   -- This reads as: "X depends on Y"
                   Y => X)
                 -- This reads as: "Y depends on X"
is
  Tmp : Integer;
begin
  X := X * 2;
  Y := Y * 2;
  Tmp := X;
  X := Y;
  X := Tmp;
  -- Oops, I mistyped... (should be "Y := Tmp;")
end Double_And_SWAP;
```

When the tools analyze the above code, they complain that the depends annotation does not match the implementation. Both variables depend on themselves instead of each other. At this point, to make the error go away we have to either change the code, or change the dependency relation. In this particular example the problem lies with the code. However, this might not always be the case, it could very well be that our contracts/specifications were actually wrong because we failed to notice a dependency and consequently failed to capture it on the Depends aspect.

Had we not added the dependency relation, it would have been easy to miss the typo and end up with an error in our code. Spotting the error was easy on this occasion but the more complicated the code the harder it gets. The tools make our life easier by highlighting the path in the code that leads to the discrepancy.

The "Plan first, act later!" advice, seems to be applicable here since programmers should first formulate their Dependency relations and then proceed to the implementation.

Lets now point out some key characteristics of aspect Depends. The Depends aspect tells us how the outputs of a subprogram relate to the inputs. Inputs always remain unchanged, so they cannot depend on anything. If an output 'X' does not depend on any input, then we have to explicitly state this by writing "Depends => (X => null)". Similarly, if an input 'Y' of the subprogram is not used by any output, we have to state this by writing "Depends => (null => Y)".

Suppose that we want to write a procedure that takes a single parameter 'Y' and then sleeps for 'Y' milliseconds. Since time is not modelled in SPARK, this procedure will appear to have no output and input 'Y' will appear to be doing nothing. The dependency relation of this Sleep procedure will look exactly as mentioned before "Depends => (null => Y)".

Lets now try to do the inverse. We will look at an unannotated piece of code and try to figure out what the corresponding Depends aspect should have been.

```
procedure No_Depends is
begin
  if Condition then
    X := Y;
  end if;
end No_Depends;
```

So let me think out loud... Since global variable Condition and global variable Y are only being read, they are inputs. Global variable X on the other hand is only ever written, so it has to just be an output. So the first Dependency relation that pops into mind is "Depends => (X => (Y, Condition))". Right?

...

WRONG ! When Condition is False, X remains exactly the way it was. So X depends on itself and is in fact also an input. It is as if we had written:

```
procedure No_Depends is
begin
  if Condition then
    X := Y;
  else
    X := X;
  end if;
end No_Depends;
```

The correct dependency for the code above would be "Depends => (X => (X, Y, Condition))". A shorthand for this is "Depends => (X =>+ (Y, Condition))". The '+' symbol means that variables on the left hand side also depend on themselves. So, the thing to remember here is that even though calculating the dependency relation of a subprogram is not too hard, there are some subtleties involved.

Aspect Depends tells us how the outputs of a subprogram relate to its inputs. This improves readability and maintainability of the code by strengthening the interface specification of a subprogram. In certain contexts, such as the development of secure systems, this is a very powerful verification/assurance technique. Here, it is recommended that programmers provide dependency relations before they start writing the actual code so that the tools can verify the validity of the implementation against the annotations. If we all were to adopt this habit, higher quality code would be generated and the world would be a better and more secure place!

National Ada Organizations

Ada-Belgium

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

Ada in Denmark

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

Ada-Deutschland

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

Ada-France

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

Ada-Spain

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

Ada in Sweden

attn. Rei Stråhle
Rimbogatan 18
SE-753 24 Uppsala
Sweden
Phone: +46 73 253 7998
Email: rei@ada-sweden.org
URL: www.ada-sweden.org

Ada-Switzerland

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