Contents

Editorial Policy for Ada User Journal 188
Editorial 189
Quarterly News Digest 190
Conference Calendar 197
Forthcoming Events 205

Ada-Europe 2019 Industrial Presentations

R. Bramberger, H. Martin, B. Gallina, C. Schmittner
“Co-engineering of Safety and Security Life Cycles for Engineering of Automotive Systems” 210

Proceedings of the "Workshop on Challenges and New Approaches for Dependable and Cyber-Physical Systems Engineering" of Ada-Europe 2019

J. Górski
“Using Evidence-Based Arguments to Support Dependability Assurance – Experiences and Challenges” 219

T. Naks, M. A. Aiello, S. T. Taft
“Using SPARK to Ensure System to Software Integrity: A Case Study” 226

J. J. Valls, M. García-Gordillo, S. Sáez
“Scenario-Based Validation & Verification, the ENABLE-S3 Approach” 230

R. Tomar, Sarishma
“Maintaining Trust in VANETs Using Blockchain” 236

Article

P. Neto, J. Tojal, J. Veríssimo, S. Melo de Sousa
“Towards a Formally Verified Space Mission Software Using SPARK” 243

Ada-Europe Associate Members (National Ada Organizations) 248

Ada-Europe Sponsors Inside Back Cover
Quarterly News Digest

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

10th Ada Developer Room at FOSDEM 2020

From: dirk@orka.cs.kuleuven.be.
(Dirk Craeynest)
Subject: CIP - Ada Developer Room at FOSDEM 2020, Brussels, Belgium
Date: Sat, 26 Oct 2019 06:43:59 -0000
Newsroups: comp.lang.ada, fr.comp.lang.ada

[NB: The call for contributions has ended but it is kept for reference. The program is given in the Forthcoming Events section of this AUJ issue --arm]

Call for Presentations
10th Ada Developer Room at FOSDEM 2020
Saturday 1 February 2020, Brussels, Belgium
Organized in cooperation with Ada-Europe

Ada-Belgium [1] is pleased to announce there will be a one-day Ada Developer Room on Saturday 1 February 2020 at FOSDEM 2020 in Brussels, Belgium. Our 10th Ada DevRoom is once more organized in cooperation with Ada-Europe [2].

General Information
FOSDEM [3], the Free and Open source Software Developers’ European Meeting, is a free and non-commercial two-day weekend event organized each year in Brussels, Belgium. It is highly developer-oriented and brings together 8000+ participants from all over the world.

No registration is necessary.

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

Ada Programming Language and Technology
Awareness of safety and security issues in software systems is ever increasing. Multi-core platforms are now abundant. These are some of the reasons that the Ada programming language and technology attracts more and more attention, among others due to Ada’s support for programming by contract and for multi-core targets. The latest Ada language definition was updated early 2016. Work on new features is ongoing, such as improved support for fine-grained parallelism, and will result in a new Ada standard scheduled for 2021. Ada-related technology such as SPARK provides a solution for the safety and security aspects stated above.

More and more tools are available, many are open source, including for small and recent platforms. Interest in Ada keeps further increasing, also in the open source community, and many exciting projects have been started.

Ada Developer Room
FOSDEM is an ideal fit for an Ada Developer Room. On the one hand, it gives the general open source community an opportunity to see what is happening in the Ada community and how Ada technology can help to produce reliable and efficient open source software. On the other hand, it gives open source Ada projects an opportunity to present themselves, get feedback and ideas, and attract participants to their project and collaboration between projects.

At previous FOSDEM events, the Ada-Belgium non-profit organization organized successful Ada Developer Rooms, offering a full day program in 2006 [4], a two-day program in 2009 [5], and full day programs in 2012-2016 [6-10], and in 2018-2019 [12]. An important goal is to present exciting Ada technology and projects also to people outside the traditional Ada community.

Our proposal for another dedicated Ada DevRoom was accepted, and now work continues to prepare the detailed program. We most probably will have a total of 8.5 schedulable hours between 10:30 and 19:00 in one of the rooms which accommodate from 59 to 85 participants.
More information will be posted on the dedicated web-page on the Ada-Belgium site [13], and final announcements will of course also be sent to various forums, lists and newsgroups.

Call for Presentations

We would like to schedule technical presentations, tutorials, demos, live performances, project status reports, discussions, etc, in the Ada Developer Room.

Ada-Belgium calls on you to:
- inform us at ada-belgium-board@cs.kuleuven.be about specific presentations you would like to hear in this Ada DevRoom;
- for bonus points, subscribe to the Ada-FOSDEM mailing list [14] to discuss and help organize the details;
- for more bonus points, be a speaker: the Ada-FOSDEM mailing list is the place to be!

Do you have a talk you want to give?
Do you have a project you would like to present?

Would you like to get more people involved with your project?
We're inviting proposals that are related to Ada software development, and include a technical oriented discussion.

You're not limited to slide presentations, of course.

Be creative. Propose something fun to share with people so they might feel some of your enthusiasm for Ada!

Speaking slots are 15 or 45 minutes, plus 5 minutes for Q&A. Depending on interest, we might also have a session with lightning presentations (e.g. 5 minutes each), and/or an informal discussion session.

Note that all talks will be streamed live (audio+video) and recorded, for remote as well as later viewing of talks, (audio+video) and recorded, when rooms are full.
We'd like to put together a draft schedule by end of November. So, please act ASAP, the sooner the better, but definitely by November 25, 2019.

We look forward to lots of feedback and proposals!

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

- a short bio and photo.
See programs of previous Ada DevRooms (URLs below) for presentation examples, as well as for the kind of info we need.

- We'd like to put together a draft schedule by end of November. So, please act ASAP, the sooner the better, but definitely by November 25, 2019.

We look forward to lots of feedback and proposals!

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

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: Aug 23, 2019
To: Ada User Journal readership

GitHub: 576 (+3) developers
Rosetta Code: 88 (+1) repositories
Bitbucket: 88 (+1) repositories
Sourceforge: 271 (+1) projects
Open Hub: 211 (+2) projects
Codelabs: 49 (+2) repositories
AdaForge: 8 (=) repositories

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

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: Thu May 23 2019
To: Ada User Journal readership
Ada-related Tools

Gnu Emacs Ada Mode 6.2.1

From: Stephen Leake
<stephen_leake@stephe-leake.org>
Subject: Gnu Emacs Ada mode 6.2.1 released.
Date: Mon, 19 Aug 2019 04:46:27 -0700
Newsgroups: comp.lang.ada

Gnu Emacs Ada mode 6.2.1 is now available in GNU ELPA. This is a minor feature and bug fix release.

The elisp parser is deleted.
Ada-mode now supports some simple refactor operations; convert between Object.Method and Prefix.Method (Object) syntax.
ada-mode now provides a project.el backend. 'project-find-file' does file name completion on files in the current project, using the 'uniquify-files' completion style.
To use this backend with an existing Ada mode project file:
(setq ada-project-current (make-ada-project :ada-prj-file <existing-file.prj>))
(add-to-list 'project-find-functions #"#ada-project-current)

Error correction is faster.

See the NEWS files in ~/.emacs.d/elpa/ada-mode-6.2.1 and wisi-2.2.1, or at http://www.nongnu.org/ada-mode/, for more details.
The process parser requires a manual compile step, after the normal list-package-installation:
cd ~/.emacs.d/elpa/ada-mode-6.2.1
=`build.sh

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

GNAT for LLVM

From: Lucretia
<laguest9000@googlemail.com>
Subject: Well, they kept that quiet!
Date: Mon, 30 Sep 2019 17:34:10 +0100

This shall be interesting...

From: Simon Wright
<simon@pushface.org>
Date: Mon, 30 Sep 2019 17:34:10 +0100

> This shall be interesting...

Indeed!

GNAT_LLVM is in gprrbuild in the macOS 9.1.0 build I put on Sourceforge, as well as in the macOS CE 2019.

From: Arnaud Charlet
Subject: Combining GNAT with LLVM
Date: Tue, 01 Oct 2019
URL: https://blog.adacore.com/combining-gnat-with-llvm

[What follows is an excerpt from the post at AdaCore's Blog. --arm]

At AdaCore labs, we have been working for some time now on combining the GNAT Ada front-end with a different code generator than GCC. [...] This time, we're looking at another general purpose code generator, called LLVM, in order to expand the outreach of Ada to the LLVM ecosystem (be it the compiler itself or other components such as static analysis tools).

This work-in-progress research project is called "GNAT LLVM" and is meant to show the feasibility of generating LLVM bytecode for Ada and to open the LLVM ecosystem to Ada, including tools such as KLEE, that we are also planning to work with and add Ada support for. Note that we are not planning on replacing any existing GNAT port based on GCC, so this project goes in addition rather than in replacement.

[...]

Simple Components for Ada 4.42 with JSON (RFC 7158)

From: "Dmitry A. Kazakov"
mailto:dmitry-kazakov.de>
Subject: ANN: Simple components for Ada v.4.42
Date: Mon, 16 Sep 2019 08:48:35 +0200
Newsroups: comp.lang.ada

The new version provides an implementation of JSON (RFC 7158) http://www.dmitry-kazakov.de/ada/components.htm

The JSON implementation supports both parsing and output of JSON objects. The parser allocates parts of the JSON object in a user-provided storage pool that can be an arena stack allowing both performance and safety by limiting the overall size of the object.

> Hi Dmitry,

> I would be curious to know the differences compared to what GNATCOLL.JSON provides, if you know? It would be nice if the Ada world did not have too many packages competing here.

Uh oh, I've written a JSON parser ([0]) too! Sorry. It's quite fast, only about 1K SLOC, supports Ada 2012's iterator and indexing syntax, Apache 2.0 license, but it does not handle UTF-8 yet (patches welcome though)

> Various things which I find limiting in GNATCOLL.JSON: performance is not really good because there are a lot of memory allocations

I did some benchmarking of some JSON parsers using a 110 M large .json file from [1]:

---

[1] https://www.tiobe.com/tiobe-index/
[3] 3,094 exercises across 52 languages, and "Level up your programming skills with Exorcism:

Wendy, the Exorcism:

URL: https://spectrum.ieee.org/static/
[1] https://www.tiobe.com/tiobe-index/
- Parsers.JSON choked on it (I used Parsers.Multiline_Source.Stream.IO)
- GNATCOLL JSON needs about ~ 17 seconds
- json-ada needs about ~ 3 seconds (should be ~ 1 second to be competitive with other languages, patches/advice welcome)

From: "Dmitry A. Kazakov" <mailto:dmitry-kazakov.de>
Date: Fri, 4 Oct 2019 14:15:08 +0200

> Parsers.JSON choked on it

I see. The reason why this is not parsed is > Parsers.JSON choked on it

Feel free to send comments.

Report preferably all comments to MacAda.org mailing list:
http://macada.org/macada/Contacts.html

See list archive:
https://hermes.gwu.edu/archives/gnat-ox.shtml

Enjoy, Pascal.
http://blady.pagesperso-orange.fr

AdaControl 1.21r6b
From: "J-P. Rosen" <rosen@adalang.fr>
Subject: [Ann] AdaControl 1.21r6b released
Date: Sun, 27 Oct 2019 11:57:05 +0100

AdaControl is pleased to announce the release of version 1.21r6 of AdaControl. This is a bug fix release, with a small addition to the rule Unnecessary_Use_Clause (see HISTORY).

Enjoy!

From: "J-P. Rosen" <rosen@adalang.fr>
Subject: [Ann] AdaControl v1.21r6b
Date: Wed, 30 Oct 2019 11:13:50 +0100

There was a minor glitch in the packaging of the version announced two days ago. Please download the correct version, now tagged 1.21r6b.

Sorry for the inconvenience.
http://adalang.org

Free-Ada Updated to GCC-9.x
From: Lucretia <laguest9000@gmail.com>
Subject: Free-Ada updated to GCC-9.x
Date: Wed, 6 Nov 2019 07:32:36 -0800
Newsroups: comp.lang.ada

[From the project’s website: “This is a set of build scripts to enable you to build the FSF Ada compiler with AdaCore’s GPL’d tools.” --arm]

Just an update to let you all know I’ve added gcc-9.2.0 support to free-ada. I’ve also added new packages.
https://github.com/Lucretia/free-ada
FYI, branches gcc-9.x and master are equivalent at this stage.

From: Jere <jhb.chat@gmail.com>
Date: Fri, 8 Nov 2019 10:46:23 -0800

Nice! This looks like it only builds in Linux for now? I saw some references to msys in there, but they don’t appear to always set the same settings as other host systems. I’m typically doing Ada development on msys2 on Win10 64bit, so was curious about it.

From: Lucretia <laguest9000@gmail.com>
Date: Fri, 8 Nov 2019 14:38:33 -0800

Yeah, only developed and tested on Linux for now. The aim is to get it working for other platforms, but getting this far on one platform is hard enough.

[...]

Ada Tools for VSCode
From: Lucretia <laguest9000@gmail.com>
Subject: Ann: VSCode extension - Ada Utilities
Date: Thu, 14 Nov 2019 10:10:17 -0800
Newsroups: comp.lang.ada

Over the last few days, I’ve knocked together a VSCode extension to make my life a bit easier. I think others might like it, just copy it in the extensions directory, I’ve not got it on the marketplace yet.
https://github.com/Lucretia/ada-utilities

From: Jere <jhb.chat@gmail.com>
Date: Thu, 14 Nov 2019 16:01:41 -0800

[Randy Brukardt asked in another post about what VSCode is. --arm]

VSCode is an open source text editor/minimalist IDE (think notepad++ maybe?) provided by microsoft. It has an extension API so developers can add various types of language support and utilities to it (this allows you to customize it to be more like a traditional IDE if you like or do whatever you want with it). I use it a lot for Ada development.

From: briot.emmanuel@gmail.com
Date: Sun, 17 Nov 2019 04:09:39 -0800

XNAdaLib binaries have been post on Source Forge:

Enjoy!

From: Blady <p.p11@orange.fr>
Subject: [Ann] XNAdaLib 2019 binaries for macOS High Sierra including GTKAda and more
Date: Sun, 13 Oct 2019 21:17:59 +0200
Newsroups: comp.lang.ada

This is XNAdaLib 2019 built on macOS 10.13 High Sierra for Native Quartz with GNAT Community 2019 including:
- GTKAda 19.0w mid-2019 (www.adacore.com/gtkada) with GTK+ 3.24.8 (www.gtk.org) complete,
- Glade 3.22.1 (glade.gnome.org),
- GnatColl 19.2 (github.com/Adacore/gnatcoll),
- Florist mid-2019a (github.com/Blady-Comp/Florist),
- AdaCurses 6.1 (invisible-island.net/ncurses/ncurses-Ada95.html),
- Gatec 0.5c (sourceforge.net/projects/lorenz),
- Components 4.41 (www.dmitry-kazakov.de/ada/components.htm),
- AICWL 3.21 (www.dmitry-kazakov.de/ada/aicwl.htm),
- Zanyblue 1.4.0 (zanyblue.sourceforge.net),
- PragmARC mid-2019 (pragmada.x10hosting.com/pragmarc.htm),
- GNOGA 1.5-beta (www.gnoga.com),
- AdaControl 1.21r3 (adalang.fr/fr/adacontrol.html),
- AdaDep 1.4r1 (adalang.fr/fr/composables.html),
- AdaSubst 1.6r5 (adalang.fr/fr/composables.html),
- SparFort 2.3-190822 (sparforte.com), and as side libraries:
- Template Parser 20.0,
- gtksourceview 3.24.4,
- GNUTLS 3.5.18,
- ASIS GPL 2019,
- SDL 1.2.15 et SDL_Image 1.2.12,
- GMP 6.1.2,
- make 4.2.1, NEW
- aspell 0.60.7, NEW
- wget 1.20.3, NEW
- Python 2.7.15,
- Python 3.6.8.

XNAdaLib binaries have been post on Source Forge:
Feel free to send comments.

Report preferably all comments to MacAda.org mailing list:
http://macada.org/macada/Contacts.html
See list archive:
https://hermes.gwu.edu/archives/gnat-ox.shtml

Enjoy, Pascal.
http://blady.pagesperso-orange.fr
To be perfectly honest, I have indeed lost most of my energy but my former padawan, Nicolas Boulenguez, has now been a full Debian Developer for several years and continues to update packages (even now doing the transition to gnat-9 for the next Debian release, in experimental). I am very proud of him.

I only occasionally skim this newsgroup and post even less often. But since you're mentioning me I have to express my gratitude and reassure Debian users that Ada is still going strong.

IRC there was a one-man effort to port Ada and many packages to DragonflyBSD, with the express purpose of stealing the crown jewel from Debian. So perhaps DragonflyBSD is a strong contender too nowadays.

A few months ago I held a two-afternoon workshop about cryptography for my 13-year-old son and a few of his friends. Of course we programmed in Ada on Debian.

Piece of cake for everyone involved to get GPS and GtAda running.

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Tue, 15 Oct 2019 19:09:31 -0700

> Debian? Ubuntu?

> [..]

But, why would there be a difference between Ubuntu and Debian? The former is very similar to the latter.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 16 Oct 2019 20:43:17 +0200

"Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de> writes:
> Ubuntu does their own work keeping Ada up to date?

Not that I know. All they do is import and recompile packages from Debian.

> AFAIK, Ubuntu has GCC 9, Debian is still 8.

Ubuntu is always, by definition and by construction, behind Debian. The compiler is only the foundation of the entire ecosystem of packages. And you should not look at the gcc package but at the gnat package.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Wed, 25 Sep 2019 10:21:46 -0700

> Seems there is an AI for this algorithm.

> Two interesting linux distributions with some Ada packages are:
> - fedora
> - gentoo

Gentoo is crap for Ada and is the reason why I started free-ada, so I could have Ada on Gentoo.

---

### Ada and Other Languages

#### Implementing Rust's Borrow Checked Pointers

From: Lucretia <laguest9000@googlemail.com>
Subject: Implementing Rust's borrow checked pointers
Date: Tue, 24 Sep 2019 02:05:44 -0700

I've been talking to someone on Telegraph and he was saying Ada should implement this, just wondering whether Ada could? I posed a slight change to access type specification to do this, what do people think?

**type P is restricted access X;**

Restricted in this case would mean that once assigned it cannot be re-assigned into or out of without some sort of move operation, which could be implemented as an attribute on the access type.

A : P := `Access;
B : P := `AMove; -- `A cannot no longer be used.

begin
A.all .; -- raises exception.

I don't know enough about this to put a complete proposal together, but I think I've got the basics understood.

From: Optikos <optikos@verizon.net>
Date: Tue, 24 Sep 2019 04:23:28 -0700

> A.all .; -- raises exception.

No, to be as useful as Rust's borrow checker, instead of raising exception, it needs to be a compile-time error. The compiler needs to maintain a whole-program directed graph at compile-time, not defer a detection-based localized analysis to run-time.

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Wed, 15 Oct 2019 19:09:31 -0700

> Restricted in this case would mean that once assigned it cannot be re-assigned into or out of without some sort of move operation, which could be implemented as an attribute on the access type.

A : P := `Access;
B : P := `A Move; -- `A cannot no longer be used.

begin
A.all .; -- raises exception.

I don't know enough about this to put a complete proposal together, but I think I've got the basics understood.

From: Lucretia <laguest9000@googlemail.com>
Date: Tue, 24 Sep 2019 04:23:28 -0700

> A.all .; -- raises exception.

No, to be as useful as Rust's borrow checker, instead of raising exception, it needs to be a compile-time error. The compiler needs to maintain a whole-program directed graph at compile-time, not defer a detection-based localized analysis to run-time.

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Wed, 15 Oct 2019 19:09:31 -0700

> Restricted in this case would mean that once assigned it cannot be re-assigned into or out of without some sort of move operation, which could be implemented as an attribute on the access type.

A : P := `Access;
B : P := `A Move; -- `A cannot no longer be used.

begin
A.all .; -- raises exception.

I don't know enough about this to put a complete proposal together, but I think I've got the basics understood.

From: Lucretia <laguest9000@googlemail.com>
Date: Tue, 24 Sep 2019 04:23:28 -0700

> A.all .; -- raises exception.

No, to be as useful as Rust's borrow checker, instead of raising exception, it needs to be a compile-time error. The compiler needs to maintain a whole-program directed graph at compile-time, not defer a detection-based localized analysis to run-time.

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Wed, 15 Oct 2019 19:09:31 -0700

> Restricted in this case would mean that once assigned it cannot be re-assigned into or out of without some sort of move operation, which could be implemented as an attribute on the access type.

A : P := `Access;
B : P := `A Move; -- `A cannot no longer be used.

begin
A.all .; -- raises exception.

I don't know enough about this to put a complete proposal together, but I think I've got the basics understood.

From: Lucretia <laguest9000@googlemail.com>
Date: Tue, 24 Sep 2019 04:23:28 -0700

> A.all .; -- raises exception.

No, to be as useful as Rust's borrow checker, instead of raising exception, it needs to be a compile-time error. The compiler needs to maintain a whole-program directed graph at compile-time, not defer a detection-based localized analysis to run-time.

From: Andrew Shvets <andrew.shvets@gmail.com>
Date: Wed, 15 Oct 2019 19:09:31 -0700

> Restricted in this case would mean that once assigned it cannot be re-assigned into or out of without some sort of move operation, which could be implemented as an attribute on the access type.

A : P := `Access;
B : P := `A Move; -- `A cannot no longer be used.

begin
A.all .; -- raises exception.

I don't know enough about this to put a complete proposal together, but I think I've got the basics understood.

From: Lucretia <laguest9000@googlemail.com>
Date: Tue, 24 Sep 2019 04:23:28 -0700

> A.all .; -- raises exception.

No, to be as useful as Rust's borrow checker, instead of raising exception, it needs to be a compile-time error. The compiler needs to maintain a whole-program directed graph at compile-time, not defer a detection-based localized analysis to run-time.
On Wednesday, August 7, 2019 at 5:09:06 AM UTC-4, Alejandro R. Mosteo wrote:
> Yes; concurrency safety (in a limited sense) is the side-effect, not the main point, I'd say.

I think at some point before the language was first stabilized (2015), both were actively pursued. Here are some musings from the original main developer back in 2013


[2]: https://manishearth.github.io/blog/2015/05/17/the-problem-with-shared-mutability/

It is interesting that both of them kind of hint at the idea of a large single thread program having similar challenges to a multithreaded program, at least when considering how undefined behavior, data invalidation, and data races occur.

[And here is another reply about self-referencing structures. --arm]

From: "Alejandro R. Mosteo"<alejandro@mosteo.com>

> Although, how does Rust's borrow checker assure the lack of cycles (or assure that the cyclic references are self-contained in a glob that itself has an acyclic reference count, so that the entire glob is condemned en masse)? It seems you are on your own (e.g. use weak references) to deal with these: https://doc.rust-lang.org/book/ch15-06-reference-cycles.html

But outside of mathematical formulas, can you give an example of single-character error in Ada?

From: Niklas Holsti <niklas.holsti@tidorum.invalid>

I once declared a record type that contained a component, aptly (I thought) called Address, of type System.Address. Then, in a certain statement dealing with a record object R of that type, I mistakenly wrote

R'Address

when I meant

R.Address

Silly me.

I now have a personal coding rule: never use the name Address for a record component of type System.Address. Perhaps this should be expanded to forbid using any record-attribute name as a component name; for example, Size.

From: Maciej Sobczak<see.my.homepage@gmail.com>

But outside of mathematical formulas, can you give an example of single-character error in Ada?

with Ada.Text_IO;
procedure Test is
procedure P (I : Integer) is
begin
Ada.Text_IO.Put_Line ("Killing people");
end;
procedure P (I : Float) is
begin
Ada.Text_IO.Put_Line ("Not killing people");
end;
begin
P (1.2); -- or should it be P (1.2); ?
end;

And of course we have to implement the Initialise operation for our Controlled types, right?

From: "Nasser M. Abbasi" <nma@12000.org>

That is why using named arguments is better and also more clear

P (I=>1, J=>2);
No chance to mix it up with

P (1=1.2);

From: Maciej Sobczak<see.my.homepage@gmail.com>

That is why using named arguments is better and also more clear

P (I=1, J=2);
Of course - the best way to avoid writing bad code is to write good code. But this is true in any language. What we should
expect from good languages is that bad code should be impossible, or at least writing bad code should take more effort than writing good code. And yet, what the above example shows, bad code is perfectly possible in Ada and in fact is easier - and that good code involves higher effort.

Which, ultimately, makes it more difficult for Ada to gain attention of C++ programmers, for example.

To Use or Not to Use Annex E

[After a question on the status of PolyORB, the following comment about the Annex E was made. --arm]

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Subject: Re: polyorb
Date: Tue, 13 Aug 2019 12:09:21 +0200
Newsgroups: comp.lang.ada

On 2019-08-13 10:13, tonyg wrote:

> The flexibility from the Distributed Systems Annex comes from the rigidity of the types used across the partitions. This being a natural amplification from the Ada language. Annex E is intended and good for tightly coupled static systems, e.g. a network of engine control units of a car.

In a loosely coupled system with nodes going on and off, changing or only modifying their roles and services, RPCs and types known prior to start is not a good choice. Another problem with RPC is that synchronous calls are utterly inefficient and slow. For a real-time system with a time-triggered transport calculated for the worst-case scenario this is no problem. But for most practical applications the load is unpredictable and millisecond accumulating latencies is not an option.

Regarding types, there were many attempts to bring some sort of abstract types and even OO to distributed systems, they all failed (CORBA, ASN.1 included).

This is why data distribution layers stick to some fixed set of primitive types leaving to the application to build upon them. Many have no types at all, only messages (e.g. MQTT). It is not nice but it works.

IMO, annex E’s remote types was a good start. But there is a lot of work required to make it full OO, to defining QoS things making it usable in loosely coupled applications.

I also think that current work on new concurrent programming primitives is wasting time. It must be invested into annex E which should serve both concurrent and distributed programming. The difference between a distributed and a multiple core system is not that dramatic (and shared memory architectures will likely die in some future anyway).

Custom Ranges and Predicates

From: Andrew Shvets <andrew.shvets@gmail.com>
Subject: How to best make a custom range?
Date: Mon, 4 Nov 2019 09:26:16 -0800
Newsgroups: comp.lang.ada

Let’s say I have the following code:

```
subtype Test_Char is Character range 'A' .. 'Z';
```

But what if I wanted to include ‘&’, ‘@’, ‘?’ and ‘!?’ in this custom range of characters as well? I thought of doing the following, but this obviously failed:

```
subtype Test_Char is Character range '@' .. '!';
```

Or is this impossible unless I use a different approach?

From: Shark8 <onewingedshark@gmail.com>
Subject: range of characters
Date: Mon, 4 Nov 2019 11:16:29 -0800

Try:

```
subtype Upper is Character range 'A'..'Z';
subtype Lower is Character range 'a'..'z';
subtype Symbol is Character with Static_Predicate =>
    Symbol in 'A'..'Z' | '@' | '?';
```

```
subtype Test_Character is Character with Static_Predicate =>
    Test_Character in Upper | Lower | Symbol;
```

```
Raise Constraint_Error;
```

From: Andrew Shvets <andrew.shvets@gmail.com>
Subject: using static predicates
Date: Tue, 5 Nov 2019 06:02:35 -0800

[... this is the warning that I get (one of many, but I can’t copy and paste): warning: in instantiation at a-nudira.adb:54 type “Result_Subtype” has predicates, attribute “First” not allowed]

From: Shark8 <onewingedshark@gmail.com>
Subject: Result_Subtype
Date: Tue, 5 Nov 2019 10:47:43 -0800

OK, the problem here is that the Ada language *does* discriminate between subtypes with predicates and those without -- mostly because there are arguments about how such attributes should behave. Things like if we have:

```
TYPE Digits is range 0..9;
SUBTYPE Odds is Digits with
    Static_Predicate => Odds in 1,3,5,7,9;
```

what should Odds’Pred(1) be? 0?
Constraint_Error?

what about Odds’Pos(1) should it be 0, the first item of the subtype? Or 1, the position in the parent-type?

From: AdaMagica <christ-usch.grein@t-online.de>
Date: Fri, 8 Nov 2019 08:07:22 -0800
Am Freitag, 8. November 2019 16:55:16 UTC+1 schrieb AdaMagica:

> Attributes work on the base type [...] See RM 3.5(25-27)

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 8 Nov 2019 16:28:56 -0600

> Attributes work on the base type, e.g.

> Natural’Pred(0) = -1

Right, but it’s a case-by-case thing as to whether an attribute applies to the type or to the subtype. The ones you’re talking about apply to the type (not any subtype), but First and Last apply to the subtype.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Tue, 5 Nov 2019 18:14:53 +0100

> warning: in instantiation at a-nudira.adb:54

> type “Result_Subtype” has predicates, attribute “First” not allowed

Because ordering attributes are ones that get broken either way. Outside generics the language treats them contravariant [the result is of the base subtype], which breaks ordering but keeps much of other semantics. When you pass a subtype as an actual parameter to a generic it suddenly becomes covariant [the result of the subtype], which is sometimes worse, sometimes quite impossible to implement. Ada plays safe here and just does not let you. In other cases you might not be so lucky. The language cannot deduce right semantics from the constraint. It is undecidable, incomputable etc. In short, do not do that.