ADA USER JOURNAL

Volume 37
Number 4
December 2016

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.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 issue of the Ada User Journal publishes the Proceedings of the "Workshop on Challenges and New Approaches for Dependable and Cyber-Physical System Engineering", which took place June 17, co-located with Ada-Europe 2016, in Pisa, Italy. This workshop brought together industry and research participants, for a full-day discussion on dependability and critical issues of Cyber-Physical Systems (CPS), a good complement to the already rich program of the Ada-Europe conference.

The workshop program included presentations from academia and industry, as well as a panel discussion. The proceedings reflect part of this rich content, starting with an Editorial by the organizers, followed by a set of technical papers. The first workshop paper, from a group of authors from Softeam, France, discusses the use of model-based design of CPS in the INTO-CPS European project. The next paper, by authors from Simula Research Laboratory and University of Oslo, Norway, present the approach of the U-Test-EU project for testing CPSs considering uncertainty. Afterwards, authors from CNR and Università di Firenze, Italy, explain the issues on model-checking of spatial and temporal properties of CPS, a work being done in the FET QUANTICOL project. Closing the proceedings, authors from University of Siena and Evidence, Italy, FORTH, Crete and BSC, Spain, present the main choices in the design of the interconnect of the AXIOM CPS board.

The issue also continues the publication of the contents of the industrial track of Ada-Europe 2016, with a paper by Vincent Monfort, of Systerel, France, describing a new Ada 2012 middleware for distributed hot-redundant software for railway control systems.

Finally, and as usual, the issue provides the News Digest, Calendar and Forthcoming Events sections, provided by the News and Events Editors, respectively Jacob Sparre Andersen and Dirk Craeynest. A special mention to Ada-Europe 2017, which will take place at the Palais Eschenbach, in the heart of Vienna, Austria, 12-16 June 2017: we are looking forward to see the Ada community gathering there for the usual high-quality Ada-Europe week.

Luís Miguel Pinho
Porto
December 2016
Email: AUJ_Editor@Ada-Europe.org

Proven Test Solutions for Reliable Embedded Software

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

-- Honeywell

VECTOR software

vectorcast.com

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

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

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]

High Integrity Software Conference

From: AdaCore Press Center
Date: Tue 20 Sep 2016
Subject: AdaCore and Altran to sponsor High Integrity Software Conference
URL: http://www.adacore.com/press/adacore-altran-high-integrity-software-conference/

AdaCore and Altran have announced their renewed sponsorship of the annual High Integrity Software Conference, which takes place in Bristol on 1 November 2016.

Following a successful conference in 2014 and 2015, organisers the University of Newcastle, AdaCore and Altran have once again compiled a complete technical programme (available at www.his-2016.co.uk/programme) to be delivered by prominent academic and industrial experts, covering topics including security, autonomy, standards and techniques & tools. This programme will be complemented by keynote talks discussing two of the high-integrity software community's most debated topics: Reasoning with Big Code from Dr Dino Stefano of Facebook and An Alternative Approach to DO-178 from Duncan Brown of Rolls-Royce.

The conference has gone from strength to strength since its inauguration in 2014. As well as the sponsorship of organisers AdaCore and Altran, this year it will benefit from the additional official support of key players BAE Systems and Jaguar Land Rover – a vote of confidence in the conference's important objective of sharing best practice in the context of today's increasing reliance on software in performing critical industrial tasks.

"HIS is one of the best conferences we attend globally due to the quality of the programme, the relevance of the exhibitors and the industries represented by the attendees," says Jamie Ayre, Commercial Team Lead at AdaCore. "We hope that the conference continues to grow from strength to strength with each edition."

Stuart Matthews, SPARK Product Manager at Altran UK, said: "We are delighted once again to be involved in this key annual conference for the software industry. This year, support has been added by two major UK companies – BAE Systems and Jaguar Land Rover – reinforcing the position of the conference as a focus for sharing experience and innovations in the field of trustworthy software development."

To find out more and register, visit www.his-2016.co.uk.

FOSDEM 2017

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Wed, 5 Oct 2016 21:55:57 +0200
*Subject: *No* Ada DevRoom at FOSDEM 2017!*
To: ADAFOSDEM@LS.KULEUVEN.BE

[...]

I am most disappointed to inform you that our proposal for an Ada DevRoom was *not* selected.

The FOSDEM organizers gave the following rationale: "[...] We also left out some long time participants to give other projects a chance as well this year." So unfortunately, no 8th Ada DevRoom at FOSDEM 2017...

[...]

[See also "FOSDEM 2017", AUJ 37-3, p. 125. —sparre]

Ada-related Resources

Ada on Social Media

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Mon Nov 7 2016
Subject: Ada on Social Media

Ada groups on various social media:

- LinkedIn: 2_528 members [1]
- Reddit: 937 readers [2]
- Google+: 708 members [3]
- StackOverflow: 633 followers [4]
- Freenode 81 participants [5]
- Twitter: 9 tweeters [6]

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

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

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

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

[5] #Ada on irc.freenode.net

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

[See also "Ada on Social Media", AUJ 37-3, p. 125. —sparre]

Repositories of Open Source Software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Mon Nov 7 2016
Subject: Repositories of Open Source software

GitHub: 1_022 repositories [1]
359 developers [1]
1_335 issues [1]
Rosetta Code: 629 examples [2]
30 developers [3]
1 issue [4]
Sourceforge: 252 repositories [5]
BlackDuck OpenHUB: 211 projects [6]
Bitbucket: 89 repositories [7]
OpenDO Forge: 24 projects [8]
500 developers [8]
Codelabs: 21 repositories [9]
AdaForge: 8 repositories [10]

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

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

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

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

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

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

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

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

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

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

[See also “Repositories of Open Source Software”, AUJ 37-3, p. 125. —sparre]

Ada-related Tools

Libkeccak (SHA3)

From: Daniel King <damaki.gh@gmail.com>
Date: Mon Aug 22 2016
Subject: Keccak and SHA3 algorithms in SPARK/Ada
URL: https://github.com/damaki/libkeccak

Libkeccak

This project implements the Keccak family of sponge functions and related constructions using the SPARK 2014 programming language.

libkeccak implements the following constructions:

- The Keccak-p permutation for state sizes of 25, 50, 100, 200, 400, 800, and 1600 bits (see [1] and [2]).
- The Sponge construction
- The Duplex construction
- Hash functions based on the Sponge construction
- eXtendable Output Functions (XOF) based on the Sponge construction

libkeccak also provides concrete implementations for the hash functions and XOFs described in NIST FIPS 202 (see [1]):

- Hash functions:
  - o SHA3-224
  - o SHA3-256
  - o SHA3-384
  - o SHA3-512
- XOFs:
  - o SHAKE128
  - o SHAKE256
  - o RawSHAKE128
  - o RawSHAKE256

Hash function configurations are also provided for the hash functions defined by the Keccak team which were submitted in the SHA-3 competition:

- Keccak-224
- Keccak-256
- Keccak-384
- Keccak-512

These hash functions differ from the final SHA-3 hash functions only in that the SHA-3 functions append two additional bits to each message, whereas the Keccak hash functions do not.

Example

Here's an example of calculating the SHA3-256 hash of a byte array (array of type Interfaces.Unsigned_8):

```
with Keccak.Types;
with SHA3;
function Compute_Hash(Data : in
         Keccak.Types.Byte_Array) return
         SHA3.SHA3_256.Digest_Type
is
   Ctx    : SHA3.SHA3_256.Context;
   Digest : SHA3.SHA3_256.Digest_Type;
begin
   SHA3.SHA3_256.Init (Ctx);
   SHA3.SHA3_256.Update (Ctx, Data);
   SHA3.SHA3_256.Final (Ctx, Digest);
   return Digest;
end Compute_Hash;
```

License

Libkeccak is licensed under the 3-clause BSD license.

[...]

Formal Verification

SPARK 2014 and GNATprove are used to provide proof that the implementation is free of errors such as: integer overflows, buffer overruns, use of uninitialized variables, and that all loops terminate. Nothing is provided currently to prove that the library correctly implements the algorithms according to the specifications. However, there are tests to provide assurances of the correctness of the algorithms with the Known Answer Tests. It is intended at some point to add proof that the algorithms correctly implement the specification.

Testing

Correctness of the algorithms is demonstrated using a combination of unit testing and Known Answer Tests (KAT). The Known Answer Tests comprise the bulk of the tests and they provide assurance that the algorithms are implemented correctly.

The unit tests aim to cover the cases that are not covered by the KAT, such as boundary conditions. As the project moves forwards I will experiment with replacing tests with proof.

References

[1] NIST FIPS PUB 202 - SHA-3 Standard: Permutation-Based Hash and Extendable output Functions. August 2015 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

[2] The Keccak Reference Version 3.0. January 2011 http://keccak.noekeon.org/Keccak-reference-3.0.pdf

[3] Cryptographic Sponge Functions Version 0.1. January 2011 http://sponge.noekeon.org/CSF-0.1.pdf

Request: Scanner Library/Binding

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sun, 18 Sep 2016 13:25:57 -0700
Subject: Scanner library Ada binding?
Newsgroups: comp.lang.ada

I'm scanning all the liner notes from my CDs (several hundred).

I've been using some old Epson software, but it's tedious. There are only 3 sizes of paper I need to scan, but the software doesn't provide for saving the sizes, so I have to "preview" and "drag the rect" for each change in size.

So I'd like to write some code to do that.

I found a Python TWAIN module, but the binary distribution is not compatible with any binary Python I could find, and compiling it from source requires MSVC (which I don't have).

So I'm looking for an Ada solution. A web search for "Ada TWAIN" didn't find anything relevant.

Does anyone have an Ada binding to the Windows TWAIN interface? I'm on 64 bit Windows 8.1, GNAT GPL 2016 (32 bit).

Or I could try Linux; I've got VMWare installed (haven't tried to see if that can see the scanner).

Random Fixed Point Numbers

From: Robert I. Eachus <rieachus@comcast.net>
Date: Mon, 19 Sep 2016 12:07:19 -0700
Subject: Re: Ada.Numerics.Float_Random. Generator question
Newsgroups: comp.lang.ada

> [...]

[...] First, what you seem to be asking for is a PRNG that selects from a uniform distribution of fixed point values. To do this take the generic discrete random package, and redeclare it with a fixed point type:

```
generic
   type Result_Subtype is delta <>;
package My_Fixed_Random is

   -- Basic facilities
   type Generator is limited private;
   function Random (Gen : Generator)
         return Result_Subtype;
   procedure Reset (Gen : in Generator;
                    Initiator : in Integer);
   procedure Reset (Gen : in Generator);
```

```
   -- Advanced facilities
   type State is private;
   procedure Save  (Gen : in Generator;
                    To_State : out State);
   procedure Reset (Gen : in Generator;
                    From_State : in State);
   Max_Image_Width : constant :=
      implementation-defined integer value;
   function Image (Of_State    : State)
         return String;
   function Value (Coded_State : String)
         return State;

private
   [...]
end My_Fixed_Random;
```

Now in the body, put an instance of generic discrete random fit to your fixed point type:

```
package body My_Fixed_Random is
   type Hidden is range Int64
         (Result_Subtype'First /
         Result_Subtype'Small) ..
          Int64 (Result_Subtype'Last  /
         Result_Subtype'Small);

   package Hidden_Pkg is new Ada.
      Numerics.Discrete_Random(Hidden);
```

Complete with the operations from My_Fixed_Random, multiplying by 'Small and converting to Result_Subtype where needed. All done. ;-) I wish I had gotten this package added to the Numerics section, as you can see it is about a page of very easy code. But there were lots of other issues that needed resolving, and not enough time to resolve them all.

Now for the evils that I hope you never meet. With this package, you should generate only numbers that are multiples of 'Small, and each legal value for Result_Subtype should occur with equal probability. You might have an issue with a value next below 'First, but usually if you have a fixed point type that officially has 2^n - 1 values, you will know what to do to avoid Constraint_Error. More troubling is that the language rules allow for fixed point types where the specified 'Small or 'Delta does not evenly divide one. I do think that permission is right--but if you use it, you had better know a lot about numerics.

GLOBE_3D

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Mon Sep 26 2016
Subject: GLOBE_3D - a GL 3D engine in Ada
URL: http://globe3d.sourceforge.net/

Features:

- unconditionally portable sources (one set of sources for all platforms)
- real-time rendering; fast with a 3D hardware-accelerated graphics card
- full eye movements and rotations
- displays combinations of colours, materials, textures
- transparency
- multitexturing (diffuse + specular) [NEW]
- multiple area rendering with the portal technique, e.g. for inner scenes
- collision detection
- binary space partition (BSP)
- input-output of 3D objects or groups of objects linked to each other by portals
- easy management of resources like textures (.bmp, .tga, .jpg, .gif, .png), BSP trees and objects, stored in .zip files
- screenshots (.bmp) and video captures (.avi)
- multi-view support
- vectorized geometry support

Goodies:

- randomly extruded surface generator

Tools:

- Export from Blender through Wavefront (.obj / .mtl) models and the o2g tool [NEW]
- VRML virtual world compiler, through the wrl2ada translator
- GMax / 3D Studio Max scene exporter & compiler, through the max2ada translator
- Compilation of game maps or levels from the Doom 3, Quake 4 or GTK Radiant level editors through the d3a (to Ada) translator or the d3g (to .g3d) tool.

[...]

[See also “GLOBE_3D”, AUJ 37-3, p. 126. —sparre]

Zip-Ada

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Sat, 8 Oct 2016 04:56:14 -0700
Subject: Ann: Zip-Ada v.52
Newsgroups: comp.lang.ada

Changes in '52', 08-Oct-2016:

- UnZip.Streams: all procedures have an additional (optional) Ignore_Directory parameter.
- Zip.Compress has the following new methods with improved compression: LZMA_3, Preselection_1 (replaces Preselection), Preselection_2. Preselection methods use now entry name extension and size for improving compression, while remaining 1-pass methods.

For those interested about what's happening "under the hood", LZMA.Encoding now computes an estimation of the predicted probabilities of some alternative encodings and chooses the most probable one - it gives an immediate better local compression. Sometimes the repetition of such a repeated short-run improvement has a long-run positive effect, but sometimes not - that's where it's beginning to be fun...

URL: https://sf.net/projects/unzip-ada/
[See also “Zip-Ada”, AUJ 37-3, p. 127. —sparre]

ADHCP

From: codelabs.ch
Date: Wed Oct 12
Subject: ADHCP
URL: https://codelabs.ch/adhcp/

[Version 0.4.2 —sparre]

ADHCP is an implementation of the DHCP protocol in Ada. Currently the project provides client services for DHCPv4, DHCPv6 and relay services for DHCPv4.

The ADHCP DHCPv4/DHCPv6 clients are D-Bus aware and can be used on most modern Linux distributions as replacement for other clients such as ISC’s dhclient or busybox’s udhcpc.

Licence

Copyright (C) 2011-2016 secunet Security Networks AG

Copyright (C) 2011-2016 Reto Buerki <reet@codelabs.ch>

Copyright (C) 2011-2016 Adrian-Ken Rueegsegger <ken@codelabs.ch>

Free use of this software is granted under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Download

Release version

The current release version of ADHCP is available at https://www.codelabs.ch/download/.

Verify a Release

To verify the integrity and authenticity of the distribution tarball, import the key https://www.codelabs.ch/keys/0xBB793815pub.asc and type the following command:

```
$ gpg --verify adhcp- version}.tar.bz2.sig
```

The key fingerprint of the public key (0xBB793815) is:

Key fingerprint = A2FB FF56 83FB 67D8 017B C50C F8C5 F8B5 BB79 3815

Development version

The current development version of ADHCP is available through its git repository:

```
$ git clone https://git.codelabs.ch/git/adhcp.git
```

A browsable version of the repository is also available here:

https://git.codelabs.ch/?p=adhcp.git

Build

To compile ADHCP on your system, you need to have the following software installed:

- GNAT compiler: https://www.gnu.org/software/gnat/gnat.html
- Anet: https://www.codelabs.ch/anet/
- Alog: https://www.codelabs.ch/alog/
- D_Bus/Ada: https://www.codelabs.ch/dbus-ada/

If you want to run the unit tests before installation of ADHCP (which is recommended) you furthermore need to have the following installed:

- Ahven (Test-Framework):

http://ahven.stronglytyped.org/

Testing

Before you install ADHCP services on your system, you might want to test if everything works as expected. ADHCP contains a unit test suite which can be run by entering the following command:

```
$ make tests
```

All tests should be marked with PASS behind the test name.

Installation

To install ADHCP on your system, type the following:

```
$ make DESTDIR=/ install
```

If DESTDIR is not specified, /usr/local is used.

Using adhcp_client on a Linux Desktop

This section describes the steps needed to use adhcp_client as DHCPv4 client on a Linux Desktop with NetworkManager. Since NetworkManager does not (yet) support ADHCP directly, an additional symlink and the adhcp_client_wrapper binary are needed.

The following procedure has been tested on Debian squeeze and testing:

```
# cd /sbin

# dpkg-divert --add --rename --divert
  /sbin/dhclient.real /sbin/dhclient

# ln -s /usr/local/sbin/adhcp_client_wrapper
  dhclient
```

If you choose another installation DESTDIR which is not in PATH add the following symlinks in /usr/local/sbin:

```
# ln -s $DESTDIR/sbin/adhcp_client
  adhcp_client

# ln -s $DESTDIR/sbin/adhcp_notify_dbus
  adhcp_notify_dbus
```

Then re-activate a NetworkManager connection which uses DHCP. If it does not work check syslog for error messages.

RFC conformity

The ADHCP implementation is designed to be simple and supports only essential features while still conforming to the related DHCP RFCs. The RFC compliance of the DHCPv4 implementation is documented here.

The RFC compliance of the DHCPv6 implementation is tested using the TAHI DHCPv6 Conformance Test Suite. The results can be found here.

Manual Pages

For more information about a specific ADHCP service or binary consult the manual pages:

- adhcp_client(8)
- adhcp_notify_dbus(8)
- adhcp_notify_simple(8)
- adhcp_client6(8)
- adhcp_notify6_dbus(8)
- adhcp_notify6_isc_script(8)
- adhcp_client_wrapper(8)
- adhcp_relay(8)

Simple Components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 13 Oct 2016 18:54:34 +0200
Subject: ANN: Simple Components for Ada v 4.16
Newsgroups: comp.lang.ada

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

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

This version fixes a bug in GNAT.Sockets.Server. The bug caused Constraint_Error in upon stopping a client after it failed to connect.

[See also “Simple Components (et al.)”, AUJ 37-3, p. 126. —sparre]

AdaSubst and AdaDep

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 18 Oct 2016 10:26:57 +0200
Subject: [Ann] New versions of AdaSubst and AdaDep
Newsgroups: comp.lang.ada

Two utilities offered by Adalog (GMGPL of course), that have been updated and improved after a period of (hmm) sleep. Executable versions are provided for use with Gnat GPL 2016.

AdaSubst:

A tool for making semantic substitution in Ada code. Provided functions include:

- Translate: Change identifiers project-wide, including when you move elements from one package to another. Adjusts the code considering all visibility rules, updates with and use clauses as necessary. Great for reorganizing projects.
- Unuse: remove use clauses, and turns all identifiers that were visible due to the use clauses to full names. Great if you like use clauses, but your clients don't!
- Unrepresent: remove all representation clauses from Ada code. Useful if you want to check your code with a different compiler, but representation clauses are incompatible.

AdaDep:

A tool for analysing dependencies. AdaDep tells you, for each withed unit, which elements are actually used.

Both can be downloaded from Adalog's components page:

http://www.adalog.fr/en/components.html

Enjoy!

Ada Security

From: Stephane Carrez <Stephane.Carrez@gmail.com>
Date: Sat Oct 29 2016
Subject: stcarrez/ada-security: Ada Security
URL: https://github.com/stcarrez/ada-security

Ada Security Library

[Implementation of RFC 6749 is under way.]

This Ada05 library provides some security frameworks needed by some Web applications. It allows a web application to integrate easily with Google, Yahoo!, Facebook and Google+ authentication systems. The library includes:

- An OpenID client authentication,
- An OAuth 2.0 client authentication,
- An OpenID Connect authentication framework,
- A policy based security framework to protect the resources

[...]

The package provides a simple AWS server that illustrates the OpenID and OpenConnect authentication. [...]

Before launching the demo server, you must update the 'samples.properties' file

and change the lines that contain PUT-HERE-YOUR-FACEBOOK-xxx and PUT-HERE-GOOGLE-xxx with your client ID and client secrets. These two changes are required by the OAuth and OpenID Connect framework only. [...]

Documentation:

The Ada Security sources as well as a wiki documentation is provided on: https://github.com/stcarrez/ada-security

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

Emacs Ada Mode

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Tue, 1 Nov 2016 14:14:26 -0700
Subject: Emacs Ada mode 5.2.1 now available
Newsgroups: comp.lang.ada

Emacs Ada mode 5.2.1 is now available in Gnu ELPA

https://savannah.nongnu.org/projects/ada-mode

http://www.nongnu.org/ada-mode/

[See also “Emacs Ada Mode”, AUJ 37-3, p. 127. —sparre]

Microsimulation Model Library

From: Graham Stark <graham.stark@virtual-worlds.biz>
Date: Tue Nov 01 2016
Subject: grahamstark/tax_benefit_model_components: Code used to build microsimulation models, mostly in the Ada language
URL: https://github.com/grahamstark/tax_benefit_model_components

Microsimulation Model Components

This is a moderately large collection of code used to build microsimulation models, mostly in the Ada language.

It includes:

- Several specialist tabulators
- An implementation of a Piecewise Linear Budget Constraint generator (see: http://www.ifs.org.uk/publications/2063)
- Interfaces to various large scale sample survey datasets;
- Utilities for building web interfaces;
- Affordability Indexes;
- Inequality Measures;
- Data Re-Weighting routines
- Tax Calculators; and
- Optimisation Routines.

There is also an experimental complete South African tax benefit model, and some, but not all, of the code required for a UK Tax Benefit model.

We've taken the liberty of bundling four useful libraries with this code. We're very grateful to the original authors:

- Strings Edit by Dmitry A.Kazakov (http://www.dmitry-kazakov.de); see http://www.dmitry-kazakov.de/ada/strings_edit.htm
- Zip-Ada by Gautier de Montmollin (http://sourceforge.net/users/gdemont/); see http://unzip-ada.sourceforge.net/
- Numeric IO by John P Woodruff (mailto:jpwoodruff@irisinternet.net); see http://www.dmitry-kazakov.de/reusable_code.htm
- XIA (XPath In Ada) by Marc A. Criley (mailto:mc@mckae.com); see http://www.mckae.com/xia.html

Contact: Graham Stark (graham.stark@virtual-worlds.biz)

See: http://virtual-worlds-research.com

Natools

From: Natasha Porté <lithiumcat@instinctive.eu>
Date: Tue Nov 1 2016
Subject: faelys/natools: Miscellaneous small utilities in Ada, gathered in one shared library
URL: https://github.com/faelys/natools

Natools

This library gather all reusable packages written by Natasha Kerensikova that are too small to fit in a project by themselves.

It contains the following package hierarchy:

- Accumulators: an interface for string accumulator objects and stacks of accumulators
  - o String_Accumulator_Linked_Lists: a basic implementation of an accumulator stack using a reference accumulator
- Chunked_Strings: an implementation of unbounded strings backed by non-contiguous fixed-size chunks of memory
- Constant_Indefinite_Ordered_Maps: task-safe ordered maps with immutable mapping
- Cron: a simple low-precision service of periodic events
- File_Streams: wrapper around Stream_IO files implementing stream interface
- Getopt_Long: command-line argument processing similar to C getopt_long
- GNAT_HMAC: instances of HMAC using GNAT hash primitives
- HMAC: generic HMAC implementation using a formal hash function
- Indefinite_Holders: simple Ada 2005 implementation of the Ada 2012 container
- References: generic simple reference-counter implementation
  - o Pools: task-safe pool of referencesjj
- S-expressions: library for dealing with S-expressions
  - o Atom_Buffers: dynamic buffer for S-expression atoms
  - o Atom_Ref_Constructors: helper constructors for atom references
  - o Atom_Refs: common reference-counted atoms
  - o Conditionals: S-expression boolean expressions about some object
    - + Generic_Evaluate: Generic boolean expression evaluation framework
    - + Strings: Boolean expressions on standard strings
  - o Dynamic_Interpreters: S-expression interpreter with mutable commands and callbacks
  - o Encodings: translators to and from official S-expression encodings
  - o Enumeration_IO: tools to help I/O of enumerations in S-expressions
  - o File_Readers: objects reading a file to an atom or a S-expression
  - o File_Writers: file-backed S-expression printer
  - o Generic_Caches: memory container for S-expressions
  - o Interpeter_Loop: inner loop of S-expression interpreters, typically used in static interpreters
  - o Interpreters: callback-based S-expressions interpreter
  - o Lockable: interface for S-expressions descriptors that can be temporarily restricted to a given nesting level
  - o Parsers: S-expression descriptor from a byte stream
  - o Printers: interface for objects to which S-expressions are written
    - + Pretty: human-friendly S-expression pretty printer
    - + Config: serialization and deserialization of pretty printer parameters to and from S-expressions
  - o Replayable: interface for S-expression descriptors whose state can be stored and replayed
  - o Special_Descriptors: always-empty and always-in-error descriptors
  - o Templates: S-expression template renderers
    - + Dates: rendering of Ada.Calendar.Time values
    - + Generic_Discrete_Render: rendering of enumeration values
    - + Generic_Integers: rendering of integer values
    - + Integers: instance of Generic_Integers for standard integers

- Static_Hash_Maps: code generator around GNAT Perfect_Hash_Generators to build a static hash map

  o S_Expressions: read S-expression description of static hash map

- Storage_Pools: helper objects with dynamic memory management

- String_Slices: objects hold slices of reference-counted shared strings

  o Slice_Sets: sets of aforementioned slices

- Tests: very simple test framework

  o Text_IO: test output using standard Ada.Text_IO

- Time_IO: conversions between time values and strings

  o Human: human-readable fuzzy formats

  o RFC_3339: time format described by RFC-3339

- Time_Keys: short printable string serialization of time that is consistent with lexicographical order

- Time_Statistics: accumulator for (run)time statistics

  o Coarse_Timers: instance of Generic_Timers with standard calendar time

  o Fine_Timers: instance of Generic_Timers using real-time annex

  o Generic_Timers: timer objects to provide data to accumulators

AdaControl

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 3 Nov 2016 16:36:05 +0100
Subject: [Ann] AdaControl 1.18r8 released
Newsgroups: comp.lang.ada

A new version of AdaControl has been released at the usual place:

http://www.adalog.fr/en/adacontrol.html

The count goes up to 68 rules and 532 subrules!

Most awaited, AdaControl accepts now directly .gpr files, and the executable versions use GNAT-GPL-2016.

See the file HISTORY for a detailed list of new features.

Enjoy!

[See also “AdaControl”, AUJ 36-1, p. 13. —sparre]

From: John Marino <dragonlace.cla@marino.st>
Date: Tue, 8 Nov 2016 07:56:49 -0800
Subject: Re: [Ann] AdaControl 1.18r8 released
Newsgroups: comp.lang.ada

> [...]

Can you confirm that gnatcoll is a now a build requirement for AdaControl?

Gnatcoll has a ton of dependencies, most of the them likely unused by AdaControl. It will be quite unfortunately if the dependency tree of AdaControl has exploded since version 17.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 8 Nov 2016 17:55:45 +0100
Subject: Re: [Ann] AdaControl 1.18r8 released
Newsgroups: comp.lang.ada

> [...]

Gnatcoll is only needed for Gnatcoll. projects (which needs gnatcoll.vfs). How do you think AdaControl is able to process project files now ;-) ?

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 10 Nov 2016 12:40:59 +0100
Subject: [Ann] AdaControl 1.18r9 released
Newsgroups: comp.lang.ada

[...]

This is a very minor release, just a small rearrangement of the sources to make it easier to compile AdaControl without Gnatcoll (but of course, you lose support of .gpr projects).

There is no change in functionality, so if you are happy with gnatcoll, there is no need to download this version.

Enjoy!

GPRBuild - A Discussion

From: Olivier Henley <olivier.henley@gmail.com>
Date: Thu, 3 Nov 2016 08:32:21 -0700
Subject: Re: Building Matreshka on Windows
Newsgroups: comp.lang.ada

Dmitry A. Kazakov wrote:

> BTW, gprbuild can C. Granted, gprbuild is a big step back from gnatmake, but still miles ahead of make.

Why do say gprbuild is a big step back from gnatmake? I don't have enough knowledge of both to evaluate your saying.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 3 Nov 2016 17:10:46 +0100
Subject: Re: Building Matreshka on Windows
Newsgroups: comp.lang.ada

> [...]

Because in my opinion it is built upon an inferior technology. It uses XML, it is not self-contained (has a mess of semi-hidden supplementary files spread all around the system). It is very easy and common to have gprbuild not working where gnatmake never had any problem. The biggest danger is turning it into yet another "configure" / CMake.

From: Simon Wright <simon@pushface.org>
Date: Thu, 03 Nov 2016 17:50:46 +0000
Subject: Re: Building Matreshka on Windows
Newsgroups: comp.lang.ada

> [...]

Unless you go out of your way, the configuration files are all under $prefix/share/gprconfig/

When things work, all is fine. When they don't, it can be tricky; the way the configuration files specify commands to run and grepping the output for particular patterns is challenging:

On the Mac, GCC 5 and 6 generate shared libraries with names like libgnat-6.dylib; GCC 4, names like libgnat-4.8.dylib. gnatls reports itself as e.g. GNATLS 6.1.0. The original share/gprconfig/compilers.xml file looked for the first 2 fields of the version; libgnat-6.1.dylib wasn't found. The fix was (cut back not to bore you):

```
   <version>
     <external>${PREFIX}gnatls -v
           --version</external>
-    <grep regexp="^GNATLS.+?(\d+(\.\d+)?)"
           group="1"></grep>
+    <grep regexp="^GNATLS.+?(\d+)"
           group="1"></grep>
   </version>I'm very pleased that, since
GPRBUILD 2016, I can use e.g.

   for Target use "arm-eabi";
   for Runtime ("ada") use
           "ravenscar-sfp-stm32f4";
```

in the GPR instead of having to specify on the command line.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 4 Nov 2016 17:08:49 +0100
Subject: Re: Building Matreshka on Windows
Newsgroups: comp.lang.ada

> [...]

You misunderstood the discussion. It was not GPR I complained about.

GPR is a language that describes an object language(s) project. I have little against it. At least it is has Ada-like syntax.

The discussion was about another language that would describe rules how to interpret a GPR file when it refers to some object language X (e.g. Ada or C). Roughly it is the language gprconfig uses.

My point was that this knowledge better be an integral part of gprbuild as it was of gnatmake. Georg suggested that somebody outside AdaCore could interested in keeping it elsewhere as grpbuild attempts to do.

[...]

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Fri, 4 Nov 2016 09:37:25 -0700
Subject: Re: Building Matreshka on Windows
Newsgroups: comp.lang.ada

> [...]

Every Ada compiler includes a tool for building a system based on the dependency information in the code.

This information is one of Ada's major strengths. One has to learn the parameters to this tool, of course, but then building becomes one line, easily stored somewhere, such as a script.

So what does a program such as gprbuild buy you? You have to tell gnatmake where to find other things used in the system (-A*), but you do the same with gprbuild, in the form of paths in the withed projects. You have to supply the compiler switches, but you do the same with gprbuild, in the Switches declarations. You don't have to write the one-line script, but instead write a many-line project file.

Personally, I don't see that gprbuild adds any value; in fact, it seems to subtract value: You supply the same information and have to have the same knowledge, but you have to know the syntax of a project file as well, so the effort is increased. For this reason, I never voluntarily use gprbuild for my projects (which are always all Ada), and don't understand why anyone would think it an advantage to do so.

(Similar arguments apply against make.)

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Mon, 07 Nov 2016 09:39:15 +0100
Subject: Re: Building Matreshka on Windows
Newsgroups: comp.lang.ada

Jeffrey R. Carter wrote:

[about all Ada projects]

> [...], easily stored somewhere, such as a script.

Yes. But don't you need to have two versions of that script, one for Microsoft hosted compilers, and one for Unix hosted compilers? Using GNAT project files, you can typically get away with having only one.

> So what does a program such as gprbuild buy you?

If your projects only contain Ada sources - nothing that "gnatmake" didn't have as well.

Since I typically work on multi-language projects (Ada, Mercurial/Subversion, Macks, C), "gprbuild" makes a big difference for me.

But I'm not at all fond of "gprconfig". I'm in the middle of figuring out how I can avoid "gprconfig", and provide ready to use ".cgpr" files instead of providing XML files for "gprconfig", which have to be installed before the project can be built.

From: Simon Wright <simon@pushface.org>
Date: Mon, 07 Nov 2016 08:50:35 +0000
Subject: Re: Building Matreshka on Windows
Newsgroups: comp.lang.ada

> [...] instead of providing XML files for "gprconfig", which have to be installed before the project can be built.

They don't have to be "installed", at any rate not in the default location. gprbuild has a --db switch:

```
--db dir Parse dir as an additional
         knowledge base
--db-    Do not load the standard
         knowledge base
```

Deepend

From: Brad Moore <bmoore.ada@gmail.com>
Date: Sun Nov 6 2016
Subject: Deepend
URL: https://sourceforge.net/projects/deepend/

Deepend is a storage pool with subpool capabilities for Ada 2005.

[...]

Released /deepend-3.5.tgz

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

Ada and Operating Systems

Linux: Joystick API

From: Artium Nihamkin <artium@nihamkin.com>
Date: Mon, 3 Oct 2016 08:54:15 -0700
Subject: ANN: Thick bindings for the Linux joystick API
Newsgroups: comp.lang.ada

Thick bindings for the Linux joystick API are available at:
https://github.com/alkhimey/Ada_Joystick

The highlight are typed axes/buttons:

```
type
   Logiteck_Extreme_3D_Pro_Axis_Type
is
  (STICK_X, STICK_Y, STICK_Z,
   THROTTLE, HAT_X, HAT_Y);
type
   Logiteck_Extreme_3D_Pro_Button_Type
is
  (BUTTON_01, BUTTON_02,
   BUTTON_03, BUTTON_04,
   BUTTON_05, BUTTON_06,
   BUTTON_07, BUTTON_08,
   BUTTON_09, BUTTON_10,
   BUTTON_11, BUTTON_12);
package L3D is new Linux_Joystick
 (Button_Type =>
   Logiteck_Extreme_3D_Pro_Button_Type,
  Axis_Type   =>
   Logiteck_Extreme_3D_Pro_Axis_Type);
```

I wrote this as an exercise, no specific use cases in mind.

If you are developing graphic application, the framework you use (GLUT, GLFW etc.) might already have joystick input functionality that is cross platform.

Mac OS X: XNAdaLib

From: Pascal Pignard <p.p11@orange.fr>
Date: Mon, 10 Oct 2016 12:37:40 -0700
Subject: [ANN] XNAdaLib 2016 binaries.
Newsgroups: comp.lang.ada

This is XNAdaLib 2016 built on Mac OS 10.11 (El Capitan) for Native Quartz including:

- GTKAda GPL 2016 with GTK+ 3.20.3 complete,
- Glade 3.18.3,
- GnatColl GPL 2016,
- Florist GPL 2016,
- AdaCurses 20110404,
- Gate 3-05-b,
- Components 4.15,
- AICWL 3.15,
- Zanyblue 1.3.0b, (new)
- PragmARC 07-2016-09, (new)
- GNOGA 1.2-beta,
- AdaControl 1.18b4, (new)
- AdaDep 1.3r3 (new)

and as side libraries:

- Template Parser,
- gtksourceview 3.14.3,
- GNUTLS 3.3.12, (new)
- ASIS GPL 2016. (new)

XNAdaLib binaries have been post on Source Forge:

https://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2016-el-capitan/

Feel free to send comments.

From: Pascal Pignard <p.p11@orange.fr>
Date: Tue, 11 Oct 2016 12:16:34 -0700
Subject: Re: [ANN] XNAdaLib 2016 binaries.
Newsgroups: comp.lang.ada

> [...]

XNAdaLib stands for extended native Ada libraries prebuilt for Mac OS.

When AdaCore set up the libre site with GPL binaries for various platforms, GTKAda was not (and is still not) shipped for Mac OS.

So I built GTKAda from source for macOS. GTKAda is based on

GTK+.Building GTK+ is long, needs lot of patches and many times I need help, so I provide the result to the community.

The library is becoming richer from year to year with AdaCore's GNATColl, Florist and many others from various contributors.

The library needs only GNAT GPL installed on macOS.

I haven't not so much feedbacks if it helps but I hope so.

[See also “Mac OS X: XNAdaLib”, AUJ 37-3, p. 128. —sparre]

Linux: Writing Kernel Modules

From: Artium Nihamkin <artium@nihamkin.com>
Date: Sun, 23 Oct 2016 06:49:14 -0700
Subject: Writing Linux Kernel Modules in Ada
Newsgroups: comp.lang.ada

I am experimenting with writing Linux kernel modules in Ada and document my attempts.

Thought some here would be interested to read about this:

http://www.nihamkin.com/2016/11/23/writing-linux-modules-in-ada-part-1/

https://github.com/alkhimey/Ada_Kernel_Module_Toolkit

Any kind of criticism is welcomed.

From: Artium Nihamkin <artium@nihamkin.com>
Date: Sat, 5 Nov 2016 15:31:01 -0700
Subject: Re: Writing Linux Kernel Modules in Ada
Newsgroups: comp.lang.ada

[...]

Part 2, incorporating the secondary stack into the run time.

http://www.nihamkin.com/2016/11/05/writing-linux-modules-in-ada-part-2/

From: Simon Wright <simon@pushface.org>
Date: Sun, 06 Nov 2016 09:39:58 +0000
Subject: Re: Writing Linux Kernel Modules in Ada
Newsgroups: comp.lang.ada

> [...]

Interesting!

What about elaboration? Have you considered the restriction No_Elaboration_Code? (would it help?)

Mac OS X: AdaSubst and AdaDep

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 25 Oct 2016 14:06:19 +0200
Subject: Adasubst and Adadep for macOS
Newsgroups: comp.lang.ada

Thanks to Pascal Pignard, a Mac OS X distribution of AdaSubst and AdaDep is available from Adalog's components page:

http://adalog.fr/en/components.html

AdaControl? will be with the next release, (hopefully) coming soon.

[See also “AdaSubst and AdaDep”, earlier in this issue. —sparre]

Windows: Compilers

From: jack <anon@example.com>
Date: Thu, 10 Nov 2016 12:50:40 -0000
Subject: Ada wanted for win7
Newsgroups: comp.lang.ada

Is there a free ada environment I can download to use on a win7-64 system?

If not I'd welcome suggestions for a low cost version.

Purpose is to learn/and evaluate Ada basics.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 10 Nov 2016 14:30:21 +0100
Subject: Re: Ada wanted for win7
Newsgroups: comp.lang.ada

> [...]

- http://www.getadanow.com/

- http://libre.adacore.com/

- http://www.adaic.org/

- http://www.rrsoftware.com/

GNAT (FSF or GPL), ObjectAda, and Janus/Ada 95 are lowest cost, TTBOMK.

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Thu, 10 Nov 2016 10:50:24 -0700
Subject: Re: Ada wanted for win7
Newsgroups: comp.lang.ada

[...]

Not addressed by others is the version of Ada you'd like to learn/evaluate. The most recent version, ISO/IEC 8652:2012, is only supported by one compiler, GNAT. Free versions of GNAT are available. GNAT also supports earlier versions of Ada.

If you want a language supported by multiple compilers you'll need to use an earlier version of Ada. The next most recent version, ISO/IEC 8652:2007, is supported by multiple vendors, but GNAT is the only free compiler that supports it.

The version of Ada before ISO/IEC 8652:2007 is ISO/IEC 8652:1995 (Ada 95), which is supported by multiple vendors. There is a free version of ObjectAda for Ada 95 that can be found on line. It has limitations as to the size of programs that it can compile, but they should not be an issue for your purposes. RR Software provides a low-cost compiler and may provide a free evaluation compiler for Ada 95.

Fedora: AVR-Ada

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Fri, 11 Nov 2016 18:35:44 +0200
Subject: AVR-Ada 1.2.2 binary rpm for Fedora 25
Newsgroups: comp.lang.ada

I created AVR-Ada 1.2.2[1] rpm for Fedora 25 (x86_64).

You can get it from my fedora.ada-language.com server, by creating a new /etc/yum.repos.d/fedora-adalanguage.repo with following contents:

[fedora-adalanguage]

name=Tero's Fedora RPM repository for Ada packages

baseurl=http://fedora.ada-language.com/repo/$releasever/$basearch

enabled=1

You can install the rpm with command:

```
sudo dnf install avr-ada --nogpgcheck
```

The rpm package is completely unofficial (not endorsed by Fedora or AVR-Ada projects), but since creating it isn't that easy, I every now and then try to provide new version.

The binaries are installed to /opt/avr-avr-122 and are usable out of the boxFor example, try:

```
export PATH=/opt/avr-avr-122/bin:$PATH
hg clone https://bitbucket.org/tkoskine/arduino-blog
cd arduino-blog/examples/hello-uart
make
```

After "make", you will have hello.hex in the current directory and you can upload it to Arduino with avrdude (not part of the package):

```
sudo avrdude -c arduino -p atmega328p -P /dev/ttyACM0 \
  -b 115200 -U flash:w:hello.hex
```

PS. Fedora 25 is not yet released, I used beta version.

[1] Well, actually the latest git version, but it is almost same.

References to Publications

Ada in Coreboot

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Mon, 19 Sep 2016 16:36:56 +0200
Subject: Ada supported in Coreboot
Newsgroups: comp.lang.ada

And as "first class citizen"!

http://www.phoronix.com/scan.php?page=news_item&px=Ada-Coreboot-First-Class

From: onox <denkpadje@gmail.com>
Date: Wed, 28 Sep 2016 06:43:24 -0700
Subject: Re: Ada supported in Coreboot
Newsgroups: comp.lang.ada

> [...]

Nice to see Ada being added to coreboot. AFAIK the Google Chromebooks use coreboot. Much leaner than Intel's UEFI, which is slow and unreliable (personal experience) IMO.

In the coreboot mailing list message that actually mention they are using SPARK. Not clear though whether that is SPARK 2014 or an older version.

From: Adrian-Ken Rueegsegger <ken@codelabs.ch>
Date: Thu, 29 Sep 2016 13:16:08 +0200
Subject: Re: Ada supported in Coreboot
Newsgroups: comp.lang.ada

> [...]

Its SPARK 2014, see e.g. the Intel graphics initialization code at [1].

[1] - https://review.coreboot.org/#/c/11869/

Ada Inside

Failure of the Schiaparelli Lander

From: Peter C. Chapin <PChapin@vtc.vsc.edu>
Date: Sat, 29 Oct 2016 09:45:13 -0700
Subject: Failure of the Schiaparelli lander: A software problem?
Newsgroups: comp.lang.ada

While not specifically Ada related, this is a topic that might be of interest to many in this group. As you may know the Schiaparelli Mars lander recently crashed to the surface of Mars instead of landing softly as intended. The article here provides some details:

http://www.skyandtelescope.com/astronomy-blogs/schiaparelli-requiem-for-a-mars-lander/

According to the article the spacecraft jettisoned its heat shield and parachute early and then, even worse, turned off the descent engines while the spacecraft was still over a mile from the ground. The article says,

"... a computer glitch seems to have confused the lander, as miscommunication between its onboard navigational system and radar erroneously told Schiaparelli it was near the surface."

I'm not sure what "computer glitch" means, exactly, but this sounds like a software failure of some kind. Does anyone here have access to more information about this? Was this a software failure? If so, what was the nature of the failure, and was it something the use of Ada (or SPARK) might have avoided?

I realize the ESA is still analyzing the telemetry from the lander so perhaps more details will come to light in time.

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Sun, 30 Oct 2016 07:19:27 -0000
Subject: Re: Failure of the Schiaparelli lander: A software problem?
Newsgroups: comp.lang.ada

> [...]

FYI, in a press release [1] earlier this year, AdaCore wrote:

"Thales Alenia Space implemented in Ada two ExoMars On-Board Software (OBSW) components: one for the Trace Gas Orbiter (TGO) on an ERC32 target, and one for the Entry, Descent and Landing Demonstrator Module (EDM) on a LEON2 target."

So there already was (at least some) Ada software on the Schiaparelli lander. The above quote seems to suggest that all software on the lander was in Ada, as the latter is named as one of the two components implemented in Ada...

Maybe someone from AdaCore or Thales on CLA can tell us more?

[1] http://www.adacore.com/press/ada-on-board-gnat-pro-helps-exomars-get-to-the-red-planet/

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Mon, 31 Oct 2016 06:48:04 -0000
Subject: Re: Failure of the Schiaparelli lander: A software problem?
Newsgroups: comp.lang.ada

> What is a "Demonstrator Module"? It is in actual control? I guess we'll have to wait.

The "Entry, Descent and Landing Demonstrator Module (EDM)" is the technical name for what is also known as the Schiaparelli lander.

<http://exploration.esa.int/mars/47852-entry-descent-and-landing-demonstrator-module/>

[See also “ExoMars”, AUJ 37-3, p. 130. —sparre]

AZip - A Portable Zip Archive Manager

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Tue, 1 Nov 2016 02:45:43 -0700
Subject: Ann: AZip v.2.0
Newsgroups: comp.lang.ada

The version 2.0 of AZip is out!

URL: http://azip.sf.net/

AZip is a Zip archive manager.

The latest addition is an archive recompression tool.

Some features:

- Flat view / Tree view
- Multi-document (should be familiar to MS Office users)
- Simple to use (at least I hope so ;-))
- Useful tools:
  - o Text search function through an archive, without having to extract files
  - o Archive updater
  - o Integrity check
  - o Archive recompression (new), using an algorithm-picking approach for improving a zip archive's compression.
- Encryption
- Methods supported: Reduce, Shrink, Implode, Deflate, Deflate64, BZip2, LZMA
- Free, open-source
- Portable (no installation needed, no DLL, no configuration file)

"Under the hood" features:

- AZip is from A to Z in Ada :-)
- Uses the highly portable Zip-Ada library
- Portability to various platforms: currently it's fully implemented with GWindows (for Windows), and there is a GtkAda draft, but anyway the key parts of the UI and user persistence are generic, platform-independent

[See also “AZip - A portable Zip Archive Manager”, AUJ 36-2, p. 69. —sparre]

AdaChess

From: Alessandro Iavicoli
Date: Tue Nov 8 2016
Subject: AdaChess is growing up: more Ada and more chess
URL: http://www.adachess.com/engine/adachess-is-growing-up-more-ada-and-more-chess.html

The upcoming AdaChess version is a really nice improvement of the previous one. AdaChess comes with some big new features both in terms of chess playing style/strength and in the Ada source code. As everybody knows, AdaChess had (has?) only two serious weakness: the speed and the search depth. Both of them has been greatly improved. Let’s see what’s new in the last release!

The engine

Compared to the last version, AdaChess is surely stronger. I will tell you soon how much it is! Those are the main changes in the engine.

The move generator

In the previous version, the generator had some bottlenecks. First of all, it played all the moves to check for legality, even moves that clearly could not leave king in check. The new generator has been designed to avoid those bottlenecks. How? AdaChess move generator try to minimize the test for legality while generating moves. It does that by looking

first if the moving piece is absolutely pinned – in this case it just moves along the line that the piece can move. On a second pass, it generate check evasion if the king is in check.

Compared to the v2.1 GSEI version, the move generator speed is 3x faster.

The search

Another important improvement is the principal variation search. The alpha-beta is still the main searching algorithm, but now there are two helper search functions: the zero-window search and the late move reduction search.

Both of them decrease the number of nodes to be searched after the “best” moves are found. Together with the more accurate score assigned to moves (to be sorted and played first) the search goes at lease 1-2 ply deeper than the v2.1 version.

Another important improvement is the use of a SEE in quiescence to avoid the engine searching for bad capture and waste time.

The hash tables

Yes! Finally I developed the hash tables for AdaChess. Thanks to the transposition tables, the engine save lot of time while searching the same positions twice or more. Currently, the hash tables works well on endgames and for recaptures where the engine saves the previous search and reuse those data.

The evaluation

In my honest opinion, the static evaluation wasn’t so bad in the 2.1 version. The only notable lack was in king defence. So, there is where I improved the evaluation by looking for attacking to the king zone and trophism.

AdaChess now full understand if each side has castled and can detect leaks in castle structure. Also, AdaChess recognize attacks to the opponent king and defends according to the weight of the attack.

Other stuffs

AdaChess has many other minor changes:

Recognize if a moves checks the opponent king and it recognize what kind of check it is;

New input/output notation. AdaChess recognize and works with your favourite notation:

Winboard/UCI: c5d6, e1g1, d7d8q, …

SAN: Bf5, Nb6+, O-O, Qxf8#, …

Long Algebraic: Nf1-g3, Rh1xd1, …

You can switch between notation by starting the engine and passing the notation as parameter with

```
adachess -n
<winboard|uci|san|long_algebraic>
```

Please note that AdaChess does not communicate by using UCI protocol. It still needs Winboard protocol.

However, AdaChess recognize all of those notations as input so if you want to build your own book just build it as you want and call it adachess.book!

Source code: more Ada and less chess

When I started to develop AdaChess I had two things in mind: making a nice chess engine in Ada. Wait, I said two things, which one is the other? Yes: making a nice chess engine and make it in Ada. It is the same sentence, one from the chess point of view and one from the Ada coding point of view. From the chess point of view it means making the engine stronger and faster. From the Ada point of view it means write code that is readable and as much idiomatic as possible.

The new code is very much Ada idiomatic and it is well commented so everybody who wants to read it can find full explanations on data structures, procedures as well as any other thing inside AdaChess. I believe that this release is very much Ada-like and I’m sure you will find the code interesting and nothing to complain about!

You can download the latest release from the download page!

AdaChess is a GPL software so feel free to use and redistribute it under the terms of the GNU Public License. If you find any bug or if you thing that can be improved in some way please contact me!

Have fun and enjoy!

[See also “AdaChess”, AUJ 34-1, p. 17. —sparre]

Ada in Context

An Obscure Arithmetic Run-Time Error

From: Yannick Moy
Date: Thu Sep 22 2016
Subject: SPARK 2014: The Most Obscure Arithmetic Run-Time Error Contest
URL: http://www.spark-2014.org/entries/detail/the-most-obscure-arithmetic-run-time-error-contest

Let's start with the typical examples of overflows and division by zero:

```
Z := X + Y;
Z := X / Y;
```

Here, X, Y and Z might be integers, fixed-point numbers or floating-point numbers. If X and Y are too big, X + Y won't fit into a machine integer or floating-point, hence the overflow error. If Y is zero, X / Y has no meaningful value, hence the division by zero error.

Let's look now at a more exotic overflow, which happens when you negate a signed integer:

```
Z := -X;
```

In the specific case where X is the minimal machine integer of this size (say, -2147483648 for a 32 bits signed integer), negating it results in a value that's one more than the maximal integer of that size (2147483647 for a 32 bits signed integer). This is because machine signed integers are asymmetric: there is one more negative value than there are positive values.

This error is rather common, because it is such a special case, so programmers tend to overlook it. Take for example the original C code of the Crazyflie small drone that we translated into SPARK. They discovered the bug after debugging a scenario that causes the drone to spin uncontrollably.

The same reason explains why there might be an overflow on absolute value operation:

```
Z := abs X;
```

Indeed, if X is -2147483648 here, its absolute value would be 2147483648 which is more than the maximal 32 bits machine integer.

There is an even more obscure overflow, which happens when you divide integers:

```
Z := X / Y;
```

In the specific case where X is again the minimal machine integer of this size (-2147483648 for a 32 bits signed integer), and Y is -1, dividing X by Y is the same as negating X, so we are back to the previous run-time error.

Now, let's open the curtain on the most obscure arithmetic run-time error: the possibility that an exponentiation on a floating-point value results in a division by zero. Yes, you may have values for X and N such that X**N results in a division by zero. How? Take X=0.0 and N=-1. The Ada standard says that X**N in that case is the same as 1/(X**(-N))... but X is zero, so X**(-N) is zero, so... yes, we have a division by zero. Amazing!

Well, it turned out that this was also a surprise for the SPARK developers. We did not know this rule of Ada, so we did not implement the check that detects this case in SPARK. This is now fixed thanks to Rod Chapman who noticed that. At this point, you should wonder how one can ever make sure to get rid of all run-time errors in her code? Are coding standard, reviews and testing going to detect all possible problems? I don't believe so. I believe that the only way to detect all possible such cases is to use a static analysis tool like SPARK. If we were able to miss one case, while we earn a living for focusing on such checking, no amount of attention and cleverness is going to make a developer catch all such cases in a real life application.

Introducing Static Analysis to Legacy Projects

From: Artium Nihamkin <artium@nihamkin.com>
Date: Sat, 5 Nov 2016 03:28:20 -0700
Subject: Re: [Ann] AdaControl 1.18r8 released
Newsgroups: comp.lang.ada

[...]

I would like to raise an issue with static analysis tools, style checkers and similar tools that is particularly valid for tools aiming at Ada projects.

Sometimes a developer decides to try these tools on an already existing codebase. This will lead to thousands of violations.

Fixing them is not acceptable in a field tested code.

The developer would like to know only about the violations in the newly added code.

Allowing this requires integration with source control tools and the ability to parse the analysis tool's log in a smart way (for example adding a line of code in the beginning of the file, will cause all the lines of the violations to increase by one)

It is interesting to hear your thoughts about this, and if you are aware of any solutions used by AdaControl users.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Sat, 5 Nov 2016 13:44:51 +0100
Subject: Re: [Ann] AdaControl 1.18r8 released
Newsgroups: comp.lang.ada

> [...]

A semantic tool, especially one using ASIS like AdaControl, can work only on compilable compilation units. OTOH, you can apply AdaControl to selected units, not necessarily a whole project.

My opinion is that if you modify a unit, then it's a good opportunity to fix any violation to coding rules. So my recommendation would be to include AdaControl in the process of entering a modified unit into the SCM.

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Mon, 07 Nov 2016 09:02:13 +0100
Subject: Re: [Ann] AdaControl 1.18r8 released
Newsgroups: comp.lang.ada

[...]

> Fixing them is not acceptable in a field tested code.

A reasonable view. - As Jean-Pierre indicates; once you touch a component, it may be time to fix observed problems in it.

We're fixing violations in field-tested components, where we estimate that it is plain luck that we haven't registered a failure due to the violation.

Cost is another argument for not fixing all violations immediately.

> The developer would like to know only about the violations in the newly added code.

Just like with compiler warnings. But how can you easily keep track? You could of course simply count the number of violations of each kind, and make sure the numbers aren't growing. How would you integrate that in a version control system? (It requires that you log analysis results for every commit, and I haven't found a nice way to do that in any of the version control systems I have used so far.)

Jean-Pierre's suggestion (to fix a file when you touch it) is tempting - until you have to change 2 lines in a 5000 line source file (with ~5000 violations). Then you quickly redefine "component" to something smaller than the whole source file - and live with not having automated checking of the rule.

From: Markus Schöpflin
Date: Mon, 7 Nov 2016 10:03:21 +0100
Subject: Re: [Ann] AdaControl 1.18r8 released
Newsgroups: comp.lang.ada

> [...]

> The developer would like to know only about the violations in the newly added code.

For a similar issue (huge legacy code base with thousands of compiler warnings) we have simply created a script which checks the output of the build against a known reference build, and then complains if the new build contains warnings not found in the reference build.

The script basically uses the following algorithm:

- For both logs, extract all warnings and remove all line number references. This results in two temporary files.
- Sort each temporary file and diff the resulting two files.
- Output all warnings found in the second file but not in the first.

After each release the reference build log is then updated.

Over the course of about five years, this has helped us to bring the warning count down from about 4000 to about 300, and to get the most critical components free of warnings.

Conference Calendar

Dirk Craeynest

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

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

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

2017

January 12-14 18th IEEE **International Symposium on High Assurance Systems Engineering** (HASE'2017), Singapore. Topics include: model-driven engineering, design languages, formal methods, domain specific languages, evolution and change, verification and validation, security and privacy, reliability and safety, tools for high assurance systems, etc. Systems of interest include: cyber-physical systems, distributed systems, embedded systems, autonomous vehicles, robot swarms, etc.

January 15-17 18th **International Conference on Verification, Model Checking, and Abstract Interpretation** (VMCAI'2017), Paris, France. Topics include: program verification, model checking, abstract interpretation, static analysis, type systems, program certification, hybrid and cyber-physical systems, etc.

☺ January 15-21 44th ACM SIGPLAN **Symposium on Principles of Programming Languages** (POPL'2017), Paris, France. Topics include: all aspects of programming languages and programming systems.

January 16-17 ACM SIGPLAN **Workshop on Partial Evaluation and Program Manipulation** (PEPM'2017). Topics include: semantics-based program manipulation; 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, test case generation, ...); techniques that treat programs/models as data objects (including: metaprogramming, generative programming, embedded domain-specific languages, model-driven program generation and transformation, ...); 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 16-20 43rd **International Conference on Current Trends in Theory and Practice of Computer Science** (SOFSEM'2017), Limerick, Ireland.

January 17-20 9th **Software Quality Days Conference** (SWQD'2017), Vienna, Austria. Topics include: improvement of software development methods and processes; testing and quality assurance of software and software-intensive systems; domain specific quality issues such as embedded, medical, automotive systems; novel trends in software quality; etc.

January 22-25 22nd IEEE **Pacific Rim International Symposium on Dependable Computing** (PRDC'2017), Christchurch, New Zealand. Topics include: architecture and system design for dependability; dependability issues in parallel and distributed systems; dependability issues in real-time systems; dependability issues in cyber-physical systems; dependability measurement, modeling, evaluation, and tools; software and hardware reliability; safety-critical systems and software; etc.

January 23-25 12th **International Conference on High Performance and Embedded Architectures and Compilers** (HiPEAC'2017), Stockholm, Sweden. Topics include: parallel, multi-core and heterogeneous systems; architectural support for programming productivity; 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.

February 01-03 — 11th **International Workshop on Variability Modelling of Software-intensive Systems** (VaMoS'2017), Eindhoven, the Netherlands. Topics include: variability across the software life cycle; architecture and design of variable software systems; formal verification, testing, and debugging of variable software systems; refactoring and evolution of variable software systems; reverse engineering of variability; formal reasoning and automated analysis on variability; software economic aspects of variability; etc.

February 05-06 — 26th **International Conference on Compiler Construction** (CC'2017), Austin, Texas, USA. Topics include: work on processing programs in the most general sense, such as compilation and interpretation techniques, run-time techniques (memory management, virtual machines, ...), programming tools (refactoring editors, checkers, verifiers, compilers, debuggers, profilers), techniques for specific domains (secure, parallel, distributed, embedded, ... environments), design and implementation of novel language constructs and programming models, etc.

February 05-07 — 10th **India Software Engineering Conference** (ISEC'2017), Jaipur, India.

February 19-21 — 5th **International Conference on Model-Driven Engineering and Software Development** (MODELSWARD'2017), Porto, Portugal. Topics include: domain-specific modeling, general-purpose modeling languages and standards, syntax and semantics of modeling languages, model-based testing and validation, model execution and simulation, model quality, component-based software engineering, software factories and software product lines, etc.

February 20-24 — 24th IEEE **International Conference on Software Analysis, Evolution, and Reengineering** (SANER'2017), Klagenfurt, Austria. Topics include: software analysis, parsing, and fact extraction; software reverse engineering and reengineering; program comprehension; software evolution analysis; software architecture recovery and reverse architecting; program transformation and refactoring; mining software repositories and software analytics; software maintenance and evolution; experience reports; education; tools and methods; etc.

Feb 27 - Mar 02 — 23rd **International Working Conference on Requirements Engineering - Foundation for Software Quality** (REFSQ'2017), Essen, Germany.

March 08-11 — 48th ACM **Technical Symposium on Computer Science Education** (SIGCSE'2017), Seattle, Washington, USA.

March 13-18 — 10th IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2017), Tokyo, Japan. Topics include: formal verification and testing, such as model checking; software reliability, security, safety, and trustworthiness; embedded software testing; testing concurrent software; testing large-scale distributed systems; testing real-time systems; testing in multi-core environments; security testing; conformance and interoperability testing; static analysis, code reviews and inspections; testing of open source and third-party software; testing and analysis tools; quality assurance; experience reports; etc. Deadline for submissions: January 12, 2017 (tool demos), January 29, 2017 (posters).

☺ April 03-06 — **The Art, Science, and Engineering of Programming Conference** (Programming'2017), Brussels, Belgium. A new conference, with an associated gold open access journal, created with the goal of placing the art of programming in the map of scholarly works. Topics include: The Art (knowledge and technical skills acquired through practice and personal experiences; examples include libraries, frameworks, languages, APIs, programming models and styles, programming pearls, and essays about programming); Science - empirical (knowledge and technical skills acquired through experiments and systematic observations; examples include user studies and programming-related data mining); Science - theoretical (knowledge and technical skills acquired through mathematical formalisms; examples include formal programming models and proofs); Engineering (knowledge and technical skills acquired through designing and building large systems and through calculated application of principles in building those systems; examples include measurements of artifacts' properties, development processes and tools, and quality assurance methods). Areas include: general-purpose programming, distributed systems programming, parallel and multi-core programming, security programming, interpreters, virtual machines and compilers, modeling and modularity, testing and debugging, program verification, programming education, programming environments, etc. Deadline for submissions: January 16, 2017 (posters).

April 03-07 32nd ACM **Symposium on Applied Computing** (SAC'2017), Marrakech, Morocco.

☺ April 03-07 **Track on Object-Oriented Programming Languages and Systems** (OOPS'2017). Topics include: aspects and components; code generation, and optimization; distribution and concurrency; formal verification; integration with other paradigms; interoperability, versioning and software evolution and adaptation; language design and implementation; modular and generic programming; runtime verification; secure and dependable software; static analysis; testing and debugging; type systems; virtual machines; etc.

☺ April 03-07 **Track on Programming Languages** (PL'2017). Topics include: compiling techniques, domain-specific languages, 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 03-07 **Track on Software Verification and Testing** (SVT'2017). Topics include: new results in formal verification and testing, technologies to improve the usability of formal methods in software engineering, applications of mechanical verification to large scale software, model checking, correct by construction development, static and run-time analysis, analysis methods for dependable systems, software certification and proof carrying code, real world applications and case studies applying software verification, etc.

April 03-07 12th **Track on Dependable and Adaptive Distributed Systems** (DADS'2017). Topics include: Dependable, Adaptive, and trustworthy Distributed Systems (DADS); middleware for DADS; modeling, design, and engineering of DADS; foundations and formal methods for DADS; etc.

April 03-07 **Track on Embedded Systems**. Topics include: design and validation of embedded systems; real-time issues; models of embedded computation; design and verification languages; operating systems and quasi-static scheduling; timing and performance analysis; power aware embedded computing; adaptive embedded systems; security in embedded systems; etc.

April 05-07 IEEE **International Conference on Software Architecture** (ICSA'2017), Gothenburg, Sweden. Topics include: model driven engineering for continuous architecting; component based software engineering and architecture design; re-factoring and evolving architecture design decisions and solutions; architecture frameworks and architecture description languages; preserving architecture quality throughout the system lifetime; software architecture for legacy systems and systems integration; architecting families of products; software architects roles and responsibilities; training, education, and certification of software architects; industrial experiments and case studies; etc. Deadline for submissions: January 10, 2017 (technical papers), February 18, 2017 (abstracts for industry track, tool papers, New and Emerging Ideas, and Young Researchers Forum), February 23, 2017 (industry track, tool papers, New and Emerging Ideas, Young Researchers Forum, workshop papers).

April 18-21 23rd IEEE **Real-Time and Embedded Technology and Applications Symposium** (RTAS'2017), Pittsburgh, PA, USA. In conjunction with CPSWeek'2017. Topics include: applications, tools, and run-time software for real-time systems; basic methodologies, algorithms, and analyses that are applied to real systems to solve specific problems; hardware/software co-design, integration methodologies, design-time tools and architectures for modern embedded systems for real-time applications; etc.

April 18-21 8th ACM/IEEE **International Conference on Cyber-Physical Systems** (ICCPS'2017), Pittsburgh, PA, USA. In conjunction with CPSWeek'2017. Topics include: security of cyber-physical systems (CPS), mechanism design for CPS, model-based design and verification of CPS, etc.

April 22-26 8th ACM/SPEC **International Conference on Performance Engineering** (ICPE'2017), L'Aquila, Italy.

April 22-29 20th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2017), Uppsala, Sweden. Events include: ESOP (European Symposium on Programming), FASE (Fundamental Approaches to Software Engineering), FOSSACS (Foundations of Software Science and Computation Structures), POST (Principles of Security and Trust), TACAS (Tools and Algorithms for the Construction and Analysis of Systems), SV-COMP (Competition on Software Verification).

April 22 — 14th **International Workshop on Formal Engineering approaches to Software Components and Architectures** (FESCA'2017). Topics include: (semi-)formal techniques and their application that aid analysis, design and implementation of software applications; formal modelling of component-based, timed and hybrid systems; temporal properties and their formal verification; interface compliance and contractual use of components; static and dynamic analysis; industrial case studies and experience reports; etc. Deadline for submissions: January 18, 2017 (paper registration), January 25, 2017 (papers).

April 26-28 — 7th **International Conference on Fundamentals of Software Engineering** (FSEN'2017), Tehran, Iran. Topics include: all aspects of formal methods, especially those related to advancing the application of formal methods in the software industry and promoting their integration with practical engineering techniques; software specification, validation, and verification; software architectures and their description languages; integration of formal and informal methods; component-based software systems; model checking; software verification; CASE tools and tool integration; industrial applications; etc.

April 28-29 — 12th **International Conference on Evaluation of Novel Approaches to Software Engineering** (ENASE'2017), Porto, Portugal. Topics include: application integration technologies, architectural design and frameworks, component-based software engineering, formal methods, model-driven engineering, reverse software engineering, software and system complexity, software and systems development methodologies, software and system quality management, software patterns and refactoring, software product line engineering, software process improvement, etc. Deadline for submissions: January 12, 2017 (workshops), January 19, 2017 (position papers), January 30, 2017 (special sessions), February 13, 2017 (special session papers), March 1, 2017 (doctoral consortium papers, open communications papers), March 6, 2017 (tutorials, demos, panels).

May 16-18 — 9th **NASA Formal Methods Symposium** (NFM'2017), Moffett Field, California, USA. Topics include: identify challenges and provide solutions for achieving assurance for critical systems; model checking; static analysis; model-based development; software and system testing; safety assurance; fault tolerance; compositional verification; design for verification and correct-by-design techniques; applications of formal methods in the development of autonomous systems, cyber-physical, embedded, and hybrid systems, ...; use of formal methods in assurance cases, automated testing and verification, ...; etc.

☺ May 20-28 — 39th **International Conference on Software Engineering** (ICSE'2017), Buenos Aires, Argentina.

May 22-23 — 20th **Ibero-American Conference on Software Engineering** (CIbSE'2017), Buenos Aires, Argentina. Event includes Software Engineering Track (SET). Deadline for submissions: January 23, 2017 (papers).

May 22-23 — 12th IEEE **International Conference on Global Software Engineering** (ICGSE'2017), Buenos Aires, Argentina. Topics include: strategic issues in distributed development, tools and infrastructure support, software architecture and design, security and privacy, lean and agile development, etc. Deadline for submissions: February 3, 2017 (Ignite talks).

May 22-26 — 16th **International Conference on Agile Software Development** (XP'2017), Cologne, Germany. Theme: "Uncovering better ways of developing software". Topics include: tools and techniques for agile development, empirical studies and evaluations, adopting and adapting agile and lean in large projects and organizations, etc.

May 29-31 — 16th **International Conference on Software Reuse** (ICSR'2017), Salvador, Brazil. Deadline for submissions: January 16, 2017 (workshops, tutorials), February 3, 2017 (Doctoral Symposium).

May 29 - Jun 02 — 31st IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2017), Orlando, Florida, USA.

♦ June 12-16 — 22nd **International Conference on Reliable Software Technologies - Ada-Europe'2017**. Vienna, Austria. Sponsored by Ada-Europe, in cooperation with ACM SIGAda, SIGBED (pending), SIGPLAN (pending), and the Ada Resource Association (ARA). Deadline for submissions: January 15, 2017 (papers, tutorials, workshops, industrial presentations).

June 12-16 — 29th **International Conference on Advanced Information Systems Engineering** (CAiSE'2017), Essen, Germany. Theme: "Digital Connected World - Informed, Disruptive Business Transformation".

Topics include: methods, models, techniques, architectures and platforms for supporting the engineering and evolution of information systems and organizations in the digital connected world. Deadline for submissions: March 20, 2017 (forum, doctoral consortium).

June 15-16 — 21st **International Conference on Evaluation and Assessment in Software Engineering** (EASE'2017), Karlskrona, Sweden. Deadline for submissions: January 22, 2017 (full paper abstracts), January 29, 2017 (full papers), March 19, 2017 (industry papers, short papers).

☺ June 19-23 — 31st **European Conference on Object-Oriented Programming** (ECOOP'2017), Barcelona, Spain. Topics include: theory, design, implementation, optimization, and analysis of programs and programming languages; innovative and creative solutions to real problems, and evaluations of existing solutions in ways that shed new insights; etc. Deadline for submissions: January 13, 2017 (papers), January 31, 2017 (workshops).

June 27-30 — 29th **Euromicro Conference on Real-Time Systems** (ECRTS'2017), Dubrovnik, Croatia. Topics include: scheduling design and analysis, real-time operating systems, hypervisors and middlewares, virtualization and timing isolation, contention-aware scheduling of multi-core systems, heterogeneous real-time systems, mixed-criticality design & assurance, WCET analysis, real-time networks and predictable communication protocols, realistic power/energy/thermal models and algorithms, network/system-on-chips and massively parallel devices, modelling and/or formal methods, industrial use-cases and RT applications, tools, compilers and benchmarks for embedded systems.

July 17-21 — **Software Technologies: Applications and Foundations** (STAF'2017), Marburg, Germany. Successor of the TOOLS federated event. Topics include: practical and foundational advances in software technology. Deadline for submissions: April 21, 2017 (workshop papers).

July 22-28 — 29th **International Conference on Computer-Aided Verification** (CAV'2017), Heidelberg, Germany. Topics include: theory and practice of computer-aided formal analysis and synthesis methods for hardware and software systems, algorithms and tools for verifying models and implementations, specifications and correctness criteria for programs and systems, deductive verification using proof assistants, program analysis and software verification, verification methods for parallel and concurrent systems, testing and run-time analysis based on verification technology, applications and case studies in verification and synthesis, verification in industrial practice, formal models and methods for security, etc. Deadline for submissions: January 24, 2017 (papers).

September 04-07 — **Federated Conference on Computer Science and Information Systems** (FedCSIS'2017), Prague, Czech Republic.

☺ Sep 12-15 — **International Conference on Parallel Computing** 2017 (ParCo'2017), Bologna, Italy. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments; new concepts for parallel computing architectures for all levels of parallelism (multicore and manycore systems, accelerators, including GPUs, FPGAs, ...); software engineering methodologies, methods and tools for developing and maintaining parallel software; parallel programming languages, compilers, libraries and environments; testing and debugging techniques and tools; best practices of parallel computing on multicore, manycore and stream processors; the application of parallel computing to solve all types of business, industrial, scientific and engineering problems using high-performance computing technologies; etc. Deadline for submissions: February 28, 2017 (extended abstracts), March 31, 2017 (mini-symposia), July 31, 2017 (full papers).

September 20-22 — 13th **International Conference on integrated Formal Methods** (iFM'2017), Turin, Italy. Topics include: hybrid approaches to formal modeling and analysis; i.e., the combination of (formal and semi-formal) methods for system development, regarding both modeling and analysis, and covering all aspects from language design through verification and analysis techniques to tools and their integration into software engineering practice. Deadline for submissions: March 28, 2017 (abstracts), April 4, 2017 (papers).

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

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

October 23-27 14th **International Colloquium on Theoretical Aspects of Computing** (ICTAC'2017), Hanoi, Vietnam.

November 07-09 30th IEEE **Conference on Software Engineering Education and Training** (CSEET'2017), Savannah, USA.

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

Conference Chair

Wolfgang Kastner
TU Vienna, Austria

Program Co-Chairs

Johann Blieberger
blieb@auto.tuwien.ac.at
TU Vienna, Austria

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

Special Session Chair

Markus Bader
markus.bader@tuwien.ac.at
TU Vienna, Austria

Tutorial and Workshop Chair

Ben Brosgol
brosgol@adacore.com
AdaCore, USA

Industrial Chair

Jacob Sparre Andersen
jacob@jacob-sparre.dk
JSA Research & innovation, Denmark

Exhibition Chair

Ahlan Marriott
ahlan@Ada-Switzerland.ch
White Elephant GmbH, Switzerland

Publicity Chair

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

Local Chair

Markus Bader
markus.bader@tuwien.ac.at
TU Vienna, Austria

General Information

The **22nd International Conference on Reliable Software Technologies – Ada-Europe 2017** will take place in Vienna, Austria. 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

15 January 2017	Submission of papers, industrial presentation, tutorial and workshop proposals
26 February 2017	Notification of acceptance to all authors
19 March 2017	Camera-ready version of papers required
30 April 2017	Industrial presentations, tutorial, and workshop material 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.

This edition of Ada-Europe features a focused **Special Session on Reliable and Safe Robotics**. Following the increasing trend of robotic systems in industrial and public environments it is more and more important to address software systems to control autonomous vehicles. This special topic discusses issues regarding challenging problems in the field of autonomous navigation and sensor fusion. Topics include (but are not limited to): **frameworks for robotics, planning and system modelling, as well as multi-agent and logistics applications**.

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

http://www.ada-europe.org/conference2017

Program Committee

Mario Aldea, *Univ. de Cantabria, Spain*
Ted Baker, *NSF, USA*
Ezio Bartocci, *Vienna University of Technology, Austria*
Bernd Burgstaller, *Yonsei Univ., Korea*
Juan A. de la Puente, *Universidad Politécnica de Madrid, Spain*
Lukas Esterle, *Vienna University of Technology, Austria*
Michael González Harbour, *Universidad de Cantabria, Spain*
J. Javier Gutiérrez, *Universidad de Cantabria, Spain*
Jérôme Hugues, *ISAE, France*
Raimund Kirner, *Univ. of Hertfordshire, UK*
Wilfried Kubinger, *FH Technikum Wien, Austria*
Albert Llemosí, *Universitat de les Illes Balears, Spain*
Kristina Lundkvist, *Mälardalen University, Sweden*
Franco Mazzanti, *ISTI-CNR, Italy*
Laurent Pautet, *Telecom ParisTech, France*
Justus Piater, *Univ. Innsbruck, Austria*
Luís Miguel Pinho, *CISTER/ISEP, Portugal*
Erhard Plödereder, *Univ. Stuttgart, Germany*
Jorge Real, *Universitat Politècnica de València, Spain*
José Ruiz, *AdaCore, France*
Sergio Sáez, *Universitat Politècnica de Valencia, Spain*
Tucker Taft, *AdaCore, USA*
Theodor Tempelmeier, *University of Applied Sciences Rosenheim, Germany*
Elena Troubitsyna, *Åbo Akademi, Finland*
Santiago Urueña, *GMV, Spain*
Tullio Vardanega, *Univ. di Padova, Italy*
Armin Wasice, *University of California at Berkeley, USA*
Michael Zillich, *Vienna University of Technology, Austria*

Industrial Committee

Ian Broster, *Rapita Systems Ltd, UK*
Jørgen Bundgaard, *Ramboll, Denmark*
Dirk Craeynest, *Ada-Belgium & KU Leuven, Belgium*
Thomas Gruber, *Austrian Institute Of Technology (AIT), Austria*
Egil Harald Høvik, *Kongsberg, Norway*
Ismael Lafoz, *Airbus Defence & Space, Spain*
Björn Lundin, *Consafe Logistics, Sweden*
Ahlan Marriott, *White Elephant GmbH, Switzerland*
Paolo Panaroni, *Intecs, Italy*
Paul Parkinson, *Wind River, UK*
Andreas Richtsfeld, *DS Automotion GmbH, Austria*
Jean-Pierre Rosen, *Adalog, France*
Emilio Salazar, *GMV, Spain*
Jacob Sparre Andersen, *JSA Research & Innovation, DK*
Jean-Loup Terraillon, *European Space Agency, The Netherlands*
Sergey Tverdyshev, *SysGO, Germany*

In cooperation with
ACM SIGAda, SIGPLAN (pending), SIGBED (pending)

acm In-Cooperation

and
Ada Resource Association (ARA)

Call for Regular and Special Session Papers

Authors of papers which are to undergo peer review for acceptance are invited to submit original contributions by 15 January 2017. Paper submissions shall not exceed 14 LNCS-style pages in length. Authors for both the general track and the special session shall submit their work via EasyChair at https://easychair.org/conferences/?conf=adaeurope2017. The format for submission is solely PDF.

The International Conference on Reliable Software Technologies is ranked class A in the CORE ranking and Microsoft Academic Search has it in the top third for conferences on programming languages. The conference is listed in DBLP, SCOPUS and Web of Science Conference Proceedings Citation index, among others.

Proceedings

Conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer, and will be available at the conference. Camera-ready accepted papers are due strictly by 19 March 2017 (format guidelines available at the conference site). Failure to comply and to register for the conference by that date will prevent the paper from appearing in the proceedings.

Call for Industrial Presentations

The conference seeks industrial presentations 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 15 January 2017. Submissions shall be made via EasyChair following the link https://easychair.org/conferences/?conf=adaeurope2017. The format for submission is solely PDF.

The Industrial Committee will review the submissions and make the selection. The authors of selected presentations shall prepare a final short abstract and submit it by 30 April 2017, 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.

Awards

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

Call for Tutorials

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

Call for Workshops

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

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.

Venue

The conference will take place at Palais Eschenbach (see images below), in the heart of Vienna, Austria.

An Invitation to Join Ada-Europe

What is Ada-Europe?

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

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

What does Ada-Europe do?

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

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

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

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

How to become a member of Ada-Europe?

Individuals

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

As benefits you will:

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

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

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

Institutions

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

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

Further information

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

National organizations contacts are available on the last page.

Ada User Journal

The journal for the international Ada community

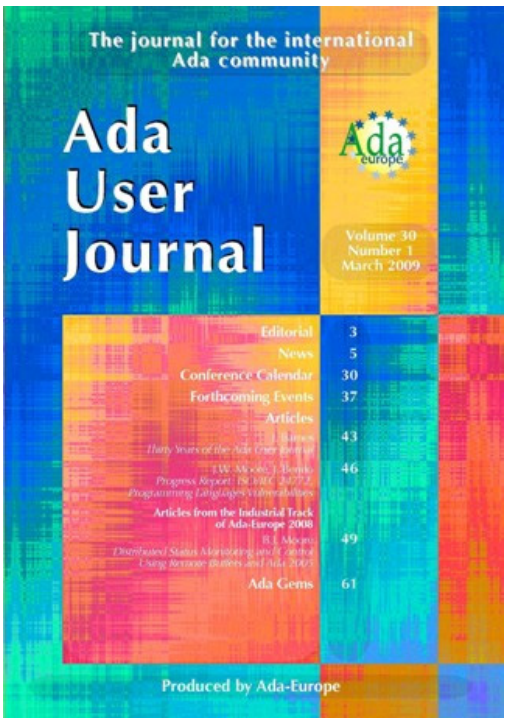
Call for Contributions

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.

Aims

The ***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 in Europe and other parts of the world. The language of the journal is English.

Although the title of the Journal refers to the Ada language, any related topics are welcome. In particular papers in areas related to reliable software technologies.



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.
- ✓ **Information** regarding standards concerning Ada.
- ✓ **Reviews** of publications in the field of software engineering.

Submission Guidelines

All material for publication should be sent to the Editor, preferably in electronic format. Prospective authors are encouraged to contact the Editor (Luís Miguel Pinho <AUJ_EDITOR@ADA-EUROPE.ORG>) to determine the best format for submission.

http://www.ada-europe.org/auj/home

Online Archive available

Middleware for a Distributed and Hot-Redundant Software in Ada 2012

Vincent Monfort
Systerel Paris, 3 Rue Danton, 92240 Malakoff; email: vincent.monfort@systerel.fr

Abstract

In railway control systems, distributed, available and reliable software is a recurrent need. As a consequence a middleware is a convenient solution to handle these aspects and provide guarantees to the application software. In this context, we developed a new Ada 2012 middleware for execution of distributed and hot-redundant software which must conform with EN50128 standard. This article presents the particular needs of such a middleware, the technical challenges and solutions, and a feedback on the Ada 2012 language and tools.

Keywords: Ada 2012, SMP, middleware, distributed software, hot-redundant software, railway control system, critical software, EN50128.

1 Introduction

For the development of its new generation of underground railway Integrated Control Center, Alstom Transportation has decided to develop a new Ada 2012 middleware after it used an Ada 83/95 one since 20 years. The reasons to develop a new middleware from scratch were the need of multi-platform and symmetric multiprocessing support on the one hand and the advantages of using the latest features of the Ada 2012 language and development tools to simplify middleware architecture on the other hand. The middleware main characteristics are a generic and high level interface to host a supervision software and to hide the mechanisms of communication, distribution (not using Annex E [1]) and hot-redundancy services provided to the application software. Moreover it must guarantee performance and high availability to the application software which in addition must conform to the EN50128 standard.

2 Context

A simplified view of an underground railway system can be described as follows, on the one side there are the wayside equipment (signaling, switches, presence sensors, etc.) and rolling stock equipment, on the other side there are the control system receiving statuses from the equipment and sending commands to the equipment. In this example (see figure 1), the control system is composed of distributed and redundant machines:

- The server is in charge to normalize status and command data, this server is duplicated on 2 redundant machines SRV1 and SRV2,
- Several control station machines (SC1, SC2 and SC3) are connected to the server (distribution).

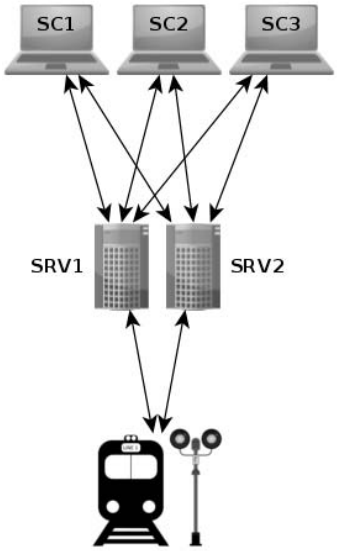


Figure 1: Example of a railway control system architecture.

In such a system, the goal of a hot-redundancy mechanism is to guarantee availability of the server services by switching from SRV1 to SRV2 (or vice versa) in a short time in case of failure of SRV1 (resp. SRV2). Each of the control station machines needs to be notified about the status changes from the wayside and can send commands to the wayside. As a consequence the middleware must reach these objectives to comply with an underground railway system needs.

3 Middleware principles

The next sections describe the main principles of the middleware which are based on the configuration of the application architecture and the execution of the application composed of application functions in different processes.

3.1 Application software architecture

In the context of the middleware, the application software architecture is composed of processes which can be executed on different machines (each process on one machine) and redundant processes which can be executed on two machines (each redundant process on two machines). Each process runs one or several application functions containing the application code of the software and which can communicate with

other determined functions. The description of this architecture must be done statically for the application, through an XML configuration file, in order for the middleware to hide the mechanisms of communication between the processes, distribution and hot-redundancy.

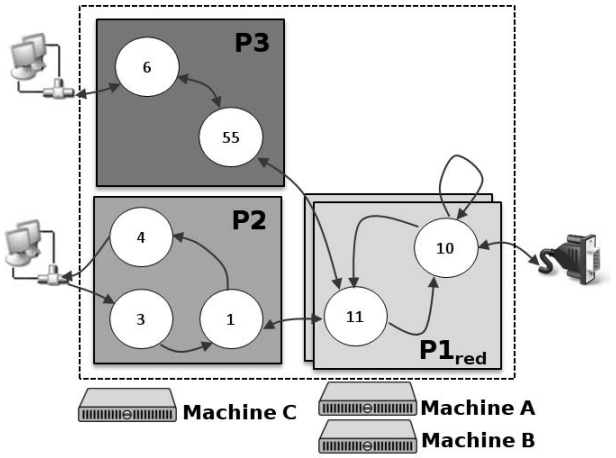


Figure 2: Example of an application architecture configuration.

Figure 2 describes an example of application architecture. It describes the machines, the processes (as squares), the application functions (as numbered circles) and configured exchanges between these functions. More particularly:

- Processes P2 and P3 are hosted on machine C: P2 runs functions 1, 3 and 4, P3 runs functions 6 and 55,
- Process P1 is redundant and then hosted on 2 machines A and B: P1 runs functions 10 and 11.

The corresponding XML configuration sample for process P1 is the following:

```
<process name = "P1">
  <hosts>
    <host name = "MachineA"/>
    <host name = "MachineB"/>
  </hosts>
  <functions>
    <function id="10"/>
    <function id="11"/>
  </functions>
  <processes_to_connect>
    <process_to_connect name = "P2"/>
    <process_to_connect name = "P3"/>
  </processes_to_connect>
</process>
```

In addition the functions can be configured to be executed on a particular CPU core with a relative priority:

```
<function id = "10" priority = "high" cpu = "1"/>
```

3.2 Execution of application functions

The execution of the application hosted by the middleware is message oriented, which means execution is based on message exchanges between functions. Since application architecture is configured, a function can send messages to (resp. receive messages from) another function without the need to take care of its process nor machine location. The exchanged messages must be defined by the application as a derived type of the abstract message type:

```
type Fid_T is range 0 .. 255; -- Function identifier type

type Message_T (From : Fid_T;
                To   : Fid_T) is abstract tagged private;

-- Application defined type for messages from function 10 to 11
type Msg_10_To_11_T is new Message_T(From => 10, To => 11)
with record ... end record;
```

A function execution is orchestrated by the reception of messages which triggers application code execution, it is then possible to execute application treatment, store data in middleware specific concurrent data structure and send messages to other functions. Each function defines the contents of the procedure, with predefined signature, called on message reception and can send messages using the predefined Send procedure:

```
-- Signature of procedure to be defined by each function
procedure Process_Message (Message : in Message_T'Class);

-- Predefined procedure to send messages
procedure Send (Message : in out Message_T);
```

An example of a function implementation can then be the following:

```
procedure Process_Message
        (Message : in Message_T'Class) is
   MyMsg : Msg_10_To_11_T; -- Declare a new message
begin
    ...                    -- Treatments of incoming message
   MyMsg.Data := ...;      -- Set the message contents to send
   MyMsg.Send;             -- Send the message
end;
```

The middleware is then in charge to deliver the message to the correct function and guarantee the delivery to the target function running in the configured process and machine.

Note: middleware guarantees are provided for a controlled and nominal application execution. It means application code does not lead to crash and can afford to treat the amount of messages it sends (if it is not the case it can be detected using a configuration option setting a limit of messages in queue).

3.3 Details of the hot-redundancy mechanism

The redundancy mechanism is mainly based on a Master/Slave mode for a redundant process, which means a redundant process instance is either active (Master mode) or passive (Slave mode) on an application functional point of view. If the Master redundant process instance has a failure, then the Slave instance will switch to Master mode and continue the treatment of messages where it was stopped. Figure 3 shows an example in which the Master instance of process P1 failed on machine A and the Slave instance switched to Master mode on machine B. In order to meet the requirements of hot-redundancy, the middleware provides the guarantee that message processing by a function is atomic and no message is processed twice or dropped in regular circumstances (nominal application behavior). These properties can be enforced thanks to a few mechanisms:

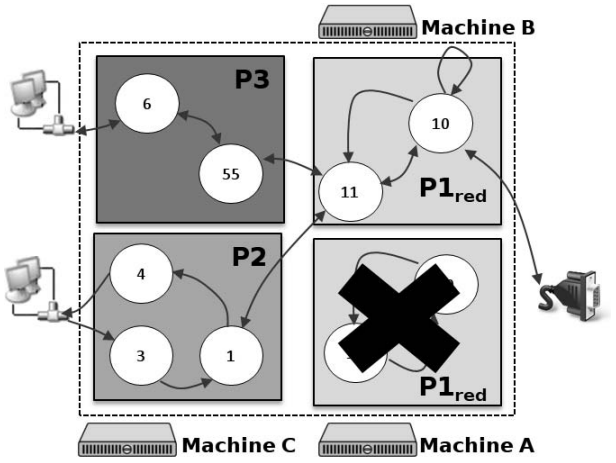


Figure 3: Example of redundant process switch from Slave to Master mode.

- Messages sent to a target function in a redundant process are transmitted to both redundant process instances,
- Messages treatment state is synchronized between the two redundant instances of a function,
- Application code must use specific redundant data structure which data are synchronized and modified in an atomic way regarding a message treatment.

As a consequence, on the application side, there are only a few constraints to implement a function in a redundant process. The application code must use only the provided redundant data structures, which are similar to classical containers such as vectors and maps, and must not store data locally or send a message referencing local contents.

4 Feedback on the Ada 2012 language and tools

This section presents feedback on the latest Ada language version and associated development tools (GNAT Pro tools). In the first place, the new middleware benefits from the use of Ada 2012 features and the first remarkable benefit is a code size reduction of about 80% compared to the precedent middleware version.

4.1 Features use

Most of the new Ada 2012 features were used to develop the middleware. The next sections describe how the new features have been used and justify the reasons for which some features were not used.

4.1.1 Multi-task and task-safe treatments

The middleware architecture makes an important use of tasks since each process uses at least the following tasks:

- client connection task: in charge to send messages from functions to the network,
- server connection task: in charge to receive messages from the network to be transmitted to target functions,
- function task: each application function execution is managed by a different task,
- state manager task: in charge of managing the redundancy state of the process.

In order to manage the messages exchanged between these tasks the *Task-safe queues* were used in combination with the *Holders* in order to store in the queues the abstract tagged ancestor message type. Indeed, the function tasks use the client queue to add messages to be sent by the client task, and the server task uses the function task queues to transmit received message to target functions.

Moreover *Synchronized barriers* are used to synchronize these tasks after initialization phase.

4.1.2 Multi-processor affinity

The *Multi-processor affinity* and *Task priority* feature are used by the middleware to configure the core tasks (client, server, etc.) and the function tasks to run on a particular CPU with a relative priority. This is important since the applications developed are intended to be used on servers with at least 8 cores.

4.1.3 Contract programming

The *Preconditions and postconditions* were widely used for the middleware development first, and then to expose the expectations and obligations of the API for the hosted application. Contract programming was really relevant to ensure quick and efficient development integration and validation.

4.1.4 Expressiveness

The use of *Conditional expressions*, *Quantified expressions* and *Expression functions*, but also of *In-Out function parameters* and *Iterators*, really improved the expressiveness and readability of the middleware code. These features contributed a lot to the code size reduction.

4.1.5 Unused features

Few of the new Ada 2012 features were not used for the middleware but we can notice the following ones.

Type invariants are not used since middleware types do not have strong internal constraints and the *Preconditions and postconditions* were widely used to express constraints between different parameters. This is certainly due to the middleware nature of the software.

Subtype predicate are also absent since only integer subtypes were defined and used only the range definition as constraint.

Ravenscar for multiprocessor systems was not used since the middleware does not use a Ravenscar profile.

In addition, as indicated in the introduction, the existing Annex E [1] was not used for distributed aspects since it was not suitable to the redundant mechanisms and encapsulation in the middleware. There was also a guarantee that we could handle the possible technical and performance issues independently to reach our goals.

4.2 Development tool use

GNAT Pro tools were used to develop in Ada 2012 and reach our requirements in terms of portability, quality and performance.

4.2.1 Operating System dependency

In order to be independent of the Operating System, the GNAT Pro libraries were used.

First of all, the GNAT sockets, used as streams, are an important feature for the middleware. Sockets are used for message exchange between processes or process instances, these messages can be application messages but also internal messages to initiate or maintain communication, synchronization of redundant process instances and so on. Associated to the stream mechanism, it permitted a lightweight and readable implementation of the message exchange on the network.

Then other Operating System dependencies were avoided through the GNAT OS_Lib interfaces.

4.2.2 Code quality

The GNATCheck tool was used to enforce the compliance of the code with the coding rules defined for the development process with EN50128 standard. Most of the coding rules were automatically verified using it.

4.2.3 Other utilities

Moreover several standard libraries were used to implement the middleware and participate also to keep the code clean and reduce its size. The XMLAda library is used to parse the application architecture configuration file, the Traceback and Source_Info libraries allow to report precise diagnostic traces and the MD5 tool is used to guarantee the integrity of the middleware version.

4.3 Difficulties encountered

After all the advantages exposed above, we also want to describe the difficulties and issues encountered during the development.

4.3.1 Development tool bugs

Several bugs showed up while using the GNAT Pro 7.2 version. They were related to the use of Ada 2012 features. Most of the bugs were compiler ones leading to crash bug box in various cases (expression function, type declaration with discriminant and derived type, anonymous type use, access to private record structure with discriminant and child unit), a few were on compiled code (non-evaluation of protected barrier, double execution of instruction) and on the GNATCheck tool (expression functions).

However all these problems have been fixed since the GNAT Pro 7.4 version.

4.3.2 Real time timers

The real time timers `Ada.Real_Time.Timing_Events` implementation is not suitable for the middleware. Indeed these timers are implemented using one task to check the expiration of all timers and execute the associated callback. Consequently, execution of one callback can postpone the following timer expiration. This last point was not acceptable since application code could have modified middleware behavior by delaying internal timers. Moreover the protected procedure callback interface was not convenient since it was forbidding to use the task-safe queues used for sending messages. Even if these timers could be suitable for embedded environment, it was not the case for the native environment and middleware needs.

As a consequence the timers were re-implemented for the middleware needs by providing non protected callback interfaces for timer expiration executed by independent tasks.

4.3.3 Performance issues

Stream sockets

Due to the combination of serialization and streaming through TCP sockets, we faced a performance issue for which the network performance was heading to 2 MBytes/sec with full CPU usage. Indeed for the serialized type, the smallest serializable sub-elements are sent as independent TCP messages which was not suitable to send our defined abstract message type. Moreover since concrete message types are defined by the application it is not suitable to let the application re-define the serialization of the type.

This problem has been solved in several steps. First we used a stream memory buffer implementation which is a stream type which can be streamed itself into another stream. Then, since it was implemented with a byte array, we had to redefine its serialization to avoid it to be sent element by element (default array serialization behavior except for the String type). Finally we used this stream as an intermediate stream to send a message type object to a TCP stream socket generating only one TCP message. This solution improved performance by a factor of 50 and brought back normal CPU usage.

Task-safe unbounded priority queues

Exchanges of messages in the middleware intensively use `Ada.Containers.Unbounded_Priority_Queues` but a performance issue was present in the implementation when using it with many messages with the same priority in the queue. Finally we participated to fix it in GNAT Pro (NF-17-OB05-042).

5 Conclusion

This middleware development was the first important industrial project using Ada 2012 for Systerel. It has convinced us that it is a major evolution of the language. The new Ada features allow for a quick development and finalization of a middleware containing a sensitive hot-redundancy feature. Finally, the first railway product application hosted by the middleware is now in production and is robust and efficient.

References

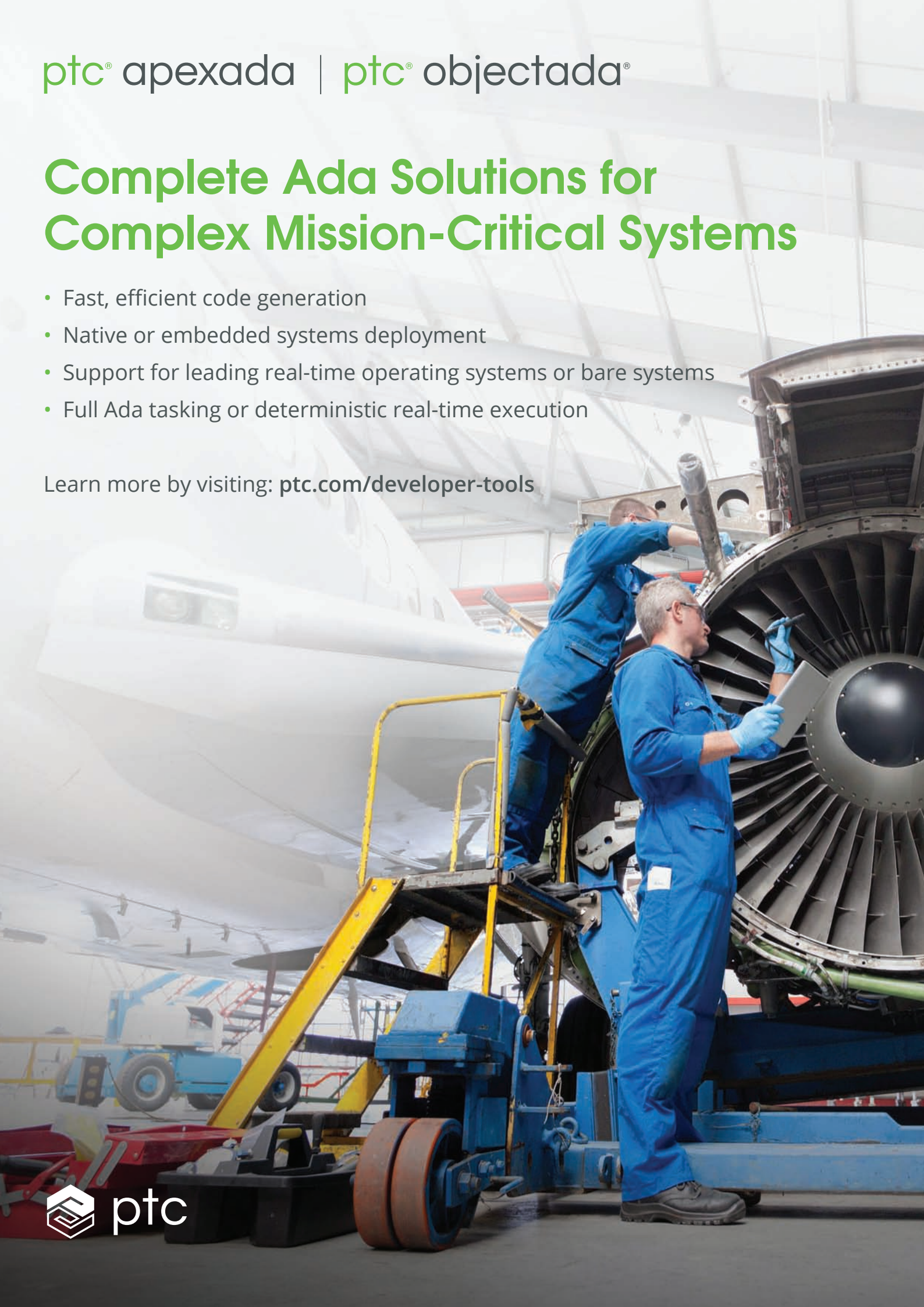
[1] ISO/IEC 8652: 2012(E), *Ada 2012 Reference Manual*.

ptc® apexada | ptc® objectada®

Complete Ada Solutions for Complex Mission-Critical Systems

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

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



ptc

Proceedings

Workshop

Challenges and New Approaches for Dependable and Cyber-Physical System Engineering (De-CPS 2016)

Ada-Europe 2016

17 June 2016

Pisa, Italy

Organizing and Program Committee

Organizers:

Silvia Mazzini, INTECS, Italy

Philippa Ryan Conmy, Adelard LLP, UK

Alessandra Bagnato, SOFTEAM, France

Daniela Cancila, CEA LIST, France

Laurent Rioux, Thales, France

Program Committee:

Katrina Attwood, University of York, UK
Huascar Espinoza, Tecnalia, Spain
Ali Koudri, IRT Systemix, France
Barbara Gallina, MDH, Sweden
Julio Medina, Universidad de Cantabria, Spain
Roberto Passerone, University of Trento, Italy
Masumi Toyoshima, DENSO, Japan
Alejandra Ruiz, Tecnalia, Spain
Silvia Mazzini, INTECS, Italy
Philippa Ryan Conmy, Adelard LLP, UK
Alessandra Bagnato, SOFTEAM, France
Daniela Cancila, CEA LIST, France
Laurent Rioux, Thales, France

Sponsors

3rd Workshop on Challenges and New Approaches for Dependable and Cyber-Physical System Engineering (De-CPS 2016) Editorial

Silvia Mazzini

Intecs, via Umberto Forti, 5 Montacchiello – I-56121 Pisa, Italy; email: silvia.mazzini@intecs.it

Daniela Cancila

CEA, LIST, CEA Saclay - F91191 Gif-sur-Yvette Cedex; email: daniela.cancila@cea.fr

Alessandra Bagnato

Softeam, 3Softeam Cadextan, 21 avenue Victor Hugo Paris 75016; email: alessandra.bagnato@softeam.fr

Philippa Ryan Conmy

Adelard LLP, 24 Waterside, 44–48 Wharf Road, London, N1 7UX, U.K.; email: pmrc@adelard.com

Laurent Rioux

THALES R&T, 1 Av. Augustin Fresnel 91767 Palaiseau Cedex – France; email: laurent.rioux@thalesgroup.com

1 The scientific view underlying De-CPS

Following the success of the inaugural workshop in 2014 and of its second edition in Madrid in 2015, in June 2016 we organized the third iteration of the workshop 'Challenges and new Approaches for Dependable and Cyber-Physical Systems engineering' (De-CPS), as satellite event of the 21th International Conference on Reliable Software Technologies – Ada-Europe 2016.

This year, the workshop focussed on:

- Industrial challenges and experience reports on co-engineering for multiple dependability concerns in Cyber-Physical Systems (CPS) engineering.
- Modeling and analysis of CPS and IoT .
- Tools and methodologies to guarantee dependability-related properties, including real-time and mixed-criticality cohabitation.
- Challenges posed for CPS design and verification by multi-core processors.
- Smart Factoring, Industry 4.0.
- Platforms for IoT - CPS.

The workshop gathers together industrial practitioners and researchers concerned with dependable CPS engineering, to foster further collaborative initiatives, and use the momentum provided by the 21th International Conference on Reliable Software Technologies to foster further collaborative initiatives.

We encourage equal representation of gender in research and innovation. Showing a strong representation by female scientists in the steering committee will hopefully inspire a new generation of women to pursue an interest in science as a career.

Several European projects accepted to contribute to the success of the workshop:

- The INTO-CPS H2020 Project: SysML for Modeling Co-Simulation Orchestration over FMI.
- The U-TEST H2020 project: Testing Cyber-Physical Systems under Realistic and Unknown Uncertainty by Combining Model and Search-Based Approaches.
- The QUANTICOL FET project: Spatio-temporal Model-checking for Collective Adaptive Systems in QUANTICOL.
- The AXIOM project: Modeling Multi-Board Communication in the AXIOM Cyber-Physical System.
- The PROXIMA FP7 project: Probabilistic real-time control of mixed-criticality multicore and manycore system.
- The ECSEL CONCERTO project. Guaranteed Component Assembly with Round Trip Analysis for Energy Efficient High-integrity Multi-core Systems.
- The ECSEL AMASS project: Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems.

- The ASSUME project: Affordable Safe & Secure Mobility Evolution.
- The EUROCPS project: Guaranteed Component Assembly with Round Trip Analysis for Energy Efficient High-integrity Multi-core Systems.

Sponsor

The workshop has been funded by CEA List.

SysML for Modeling Co-simulation Orchestration over FMI: the INTO-CPS Approach

Alessandra Bagnato, Etienne Brosse, Imran Quadri and Andrey Sadovykh

Softeam Research & Development Division; Paris, France; email : FirstName.LastName@softeam.fr

Abstract

This paper describes a work in progress related to Model-Based Design (MBD) of Cyber-Physical Systems (CPSs) in the context of the Horizon2020 INTO-CPS project. The paper highlights usage of the INTO-CPS SysML profile, that extends SysML profile, to enable defining co-simulations over Functional Mock-up Interface (FMI) in the INTO-CPS platform. The paper focuses on the results of the first year of the project.

Keywords: Model-Based Design, Cyber-Physical Systems, Co-Simulation.

1 Introduction

The Horizon2020 INTO-CPS project focuses on the "model-based design" of Cyber-Physical Systems (CPSs). A model signifies a representation of some reality or systems with an accepted level of abstraction, i.e. all unnecessary details of the system are omitted for the sake of simplicity, formality and comprehensibility, etc. INTO-CPS promotes heterogeneous modeling, as the overall model in INTO-CPS is actually a multi-model in nature, where each model can be described using a different modeling notation/language, possibly from a different computational paradigm or domain. In INTO-CPS, models are categorized into continuous time (CT) and discrete event models (DE) [1].

INTO-CPS advocates for a multi-modeling covering both holistic and design architecture approaches. A holistic architecture describes a conceptual model that highlights the main elements of the system and the way these elements are connected with each other, taking a holistic view of the overall system. The aim is to identify the main functional elements of the system reflecting the semantics and structure of the domain of application. The design architecture emphasizes a decomposition of a system into subsystems, where each subsystem is modelled separately in isolation using a special notation and tool designed for the domain of the subsystem. The INTO-CPS SysML profile is designed to enable the specification of CPS design architectures [2].

The INTO-CPS toolchain is a collection of software tools, based centrally around Functional Mock-up Interface (FMI) [3] compatible co-simulation that supports the collaborative development of CPSs. The INTO-CPS application interfaces tools in the INTO-CPS toolchain to query model information which can be used in collaborative simulations (co-simulations) by the INTO-CPS Co-Simulation Orchestration Engine (COE) [2]. It should be noted that FMI is central to the INTO-CPS toolchain [3].

Co-simulation in INTO-CPS is the simultaneous and collaborative execution of multi-models enabling complex validation scenarios that integrate cooperation of several sub-systems at the same time. The INTO-CPS multi-model may be CT-only, DE-only or a combination of both. The Co-simulation Orchestration Engine (COE) combines existing co-simulation solutions and scales them to the CPS level, allowing CPS multi-models to be evaluated through co-simulation.

The paper first describes the FMI standard, followed by the INTO-CPS model-based approach. The Modelio modeling environment is subsequently presented to enable high-level modeling of the INTO-CPS models. Afterwards relevant FMI concepts are presented and how they are mapped in the INTO-CPS SysML profile; followed by a conclusion.

2 FUNCTIONAL MOCK-UP INTERFACE (FMI)

The Functional Mock-up Interface (FMI) is a tool-independent standard to support both model exchange and co-simulation of dynamic models using a combination of XML-files and compiled C-code [4]. Part of the FMI standard for model exchange specification is a model description file, an XML file, that supplies a description of all properties of a model (for example input/output variables). A Functional Mock-up Unit (FMU) implements FMI. A FMU is in fact the executable that implements the interface. Data exchange between FMUs and the synchronization of all simulation solvers [4] is controlled by a Master Algorithm (MA). During FMU export, a FMU archive is generated from a systems model, while during FMU import a systems model is generated from a FMU archive.

An FMU contains the following information:

A. ModelDescription.xml file

The model description file, in XML format, contains information about the CPS model, for example variable definitions, attributes, and other more general model information, for e.g. model name and FMI version. The advantages of this approach are that there is no overhead for model execution and the tools independency, the tools can read this information with their preferred language (for example C++, Java, etc.).

B. Model equations

A CPS model behavior can be described for example using equations (differential, algebraic or discrete equations). These equations are represented by a small set of C functions. The source code and/or binary code for one or more platforms (such as Windows/Linux) is present in the FMU. One FMU can contain binaries for more than one platform and/or platform version.

C. Optional resource files

The FMU can contain other optional files that might be useful for the model, such as documentation files (HTML), model icon (bitmap file), maps and tables (read by model during initialization), and other libraries or DLLs that need to be used in the model.

3 FMI FOR MODEL-EXCHANGE AND CO-SIMULATION

The FMI standards currently specify two types of protocols:

1. FMI for Model Exchange (import and export).

2. FMI for Co-Simulation (master and slave).

For FMI Model Exchange import, the subsystem model is exported from a simulation tool in the form of an FMU archive containing the necessary FMU information (model description file, optional C source code, etc.); while in the FMI Model Exchange export, the subsystem model is imported into the simulation system for system simulation. The FMI for Co-Simulation is to couple two or more simulation tools in a co-simulation environment, such as the INTO-CPS COE [2].

The main difference between these two protocols is that in Model Exchange the FMU is simulated using the importing tool's solver, while in Co-Simulation the FMU is shipped with its own solver.

4 INTO-CPS MODEL-BASED APPROACH

The structure and syntax of the expressions should be formally defined in order to be interpretable by a machine. This is achieved by a metamodel. A metamodel is a collection of concepts and relations for describing a model using a model description language; and is used for defining the syntax of a model.

Each model that is designed according to a given metamodel is said to conform to its metamodel at a higher level. This relation is analogous to a text and its language grammar. Here level does not signify an abstraction level, but a definition level. A metamodel itself is also a model, thus it also conforms to another metamodel. However, in order to define a model, it is not convenient to define an infinite succession of metamodels, with each one conforming to another at a higher level. One formal solution to this issue is the definition of a metamodel, which conforms to itself, i.e., it can be expressed only by using the concepts it defines. Currently, widely used notations, such as MOF [5] and its implementation Ecore [6], are examples of such kind of metamodels or metametamodels.

UML is considered as one of the main unified visual modeling languages in Model-Based Design [7]. The UML metamodel was standardized in 1997 by the Object Management Group (OMG). Since its standardization, UML has been widely accepted and adopted in industry and academia. UML now provides support for a wide variety of modeling domains, including real-time system modeling. It has been proposed to answer the requirements of modeling specification, communication, documentation, etc. It integrates the advantages of component re-use, unified modeling of heterogeneous system, different facet modeling of a system, etc. The proposition of UML is based on several languages, such as Booch and OOSE, which had a great influence on the object-based modeling approach. Consequently, UML is very similar to object-based languages. As UML is widely utilized in industry and academia for modeling purposes, a large number of tools have been developed for its support.

Unfortunately, the success of UML has its drawbacks, resulting in a bloated and complex language. Its expressivity and precision are not always well defined in certain cases for the specification of some specific systems; such as CPSs.

System Modeling Language (SysML) [8] is the first UML standard for system engineering proposed by OMG that aims at describing complex systems. SysML allows describing of the traceability requirements, and provides means to express the behavior and composition of the system blocks.

The key contributions of the SysML standard are as follows:

Architecture organization: The modeling concepts related to expressing architectural aspects.

Blocks and Flows: Blocks allow representing complex systems in a composed manner. Flows in SysML enable modeling of data/control flow as well as physical flows, such as electrical flows.

Behavior: SysML refines the UML common behavior concepts (such as state machines, activities among others) for modeling continuous systems.

Requirements: System requirements can also be modeled via SysML. These requirements can be presented either in graphical or tabular form and help with model traceability.

Parametrics: SysML allows designers to describe analytical relations and constraints in a graphical manner.

The INTO-CPS SysML profile, defined in [2], proposes the following structuring of a CPS: Systems are decomposed into “subsystems”. These subsystems may be composed of cyber or physical components. Using the INTO-CPS tool chain, each subsystem corresponds to a FMI model description, and therefore an individual model in a multi-model.

5 MODELIO MODELING ENVIRONMENT

Modelio is an open source modeling tool [9] supporting UML standard and its extensions, such as SysML. Designers are thus able to simultaneously depict and specify several aspects of the system from requirements to the hardware/software architecture through use case specification, and system functional design. Modelio also enables transversal features such as automatic documentation generation and traceability/impact analysis which are helpful during system life cycle management. Customization mechanisms provided by Modelio allow the definition of dedicated working environments. Additionally, Modelio supports the notion of World Wide modeling, which enables model fragments to be distributed and reused in distributed working environments. Modelio allows to both define and develop the SysML profile and to carry out the automatic transformation [10, 11] of SysML models into FMI, and vice versa. The subsequent section provides details on how to model the INTO-CPS SysML models containing CPS components, in order for them to be automatically transformed into FMI, and vice versa.

6 FMI AND SYSML FOR INTO-CPS

The following section provides an example of a ModelDescription.xml file related to cascading water tank developed in 20-sim [12] and illustrates the FMI syntax.

The modelDescription.xml contains the definition of the simulation interface including input and output variables as well as some internal variables for monitoring purposes. The interfaces are defined in the XML language.

The main part of the description is the model variables definition. The FMI limits the definition to scalar variables. Thus, all the variables should be flattened and listed as a collection of scalar variables. For example, Arrays should be flattened to scalars. Each scalar variable is defined by its:

- **Name**: a unique identifier with respect to all other elements of the ModelDescription file.
- **ValueReference**: a default starting value (can be for example: Real, Integer, or Boolean).
- **Description**: a textual description of the scalar variable.
- **Variability**: An enumeration that defines the time dependency of the variable, i.e. defines the time instants when a variable can change its value: Can be Discrete, Continuous, Tunable or Fixed.
- **Discrete**: Value is constant between external and internal events (i.e. time, state, steps defined implicitly in the FMU).
- **Continuous**: no restriction on value changes (only variable of type Real can be continuous).
- **Tunable**: Value is constant between external events.
- **Fixed**: Fixed value after initialization.
- **Causality**: Enumeration that defines the causality of the variable: Parameter, Local, Input or Output.
- **Input**: The variable value is provided from another model or slave.
- **Output**: The variable value can be used by other model or slave.
- **Local**: A local scalar variable is not to be used outside of the FMU scope. Its value is calculated from other variables.
- **Parameter**: A parameter value is not changed after initialization. The value remains constant during simulation and is not used in connections to other FMUs. When the causality is set as “parameter”, the variability must be set to either “tunable” or “fixed”.

In INTO-CPS we use SysML to define the composition of the simulations that are defined with FMI interfaces. In the subsequent section, we define the mapping between INTO-CPS SysML concepts and FMI 2.0; with the intended goal to automatically generate FMI related to FMUs from an INTO-CPS SysML model and to also enable the reverse process, i.e. import an FMI related to a FMU into Modelio modeling environment; to automatically generate a corresponding INTO-CPS SysML model.

7 INTO-CPS SYSML MODELING

The section below describes the initial version of INTO-CPS approach taken from the first year of experiments. This evolution of this approach is further discussed in conclusion. Two new diagrams in the scope of INTO-CPS have been developed and are described herewith, the Architecture Structure Diagram and the Connection Diagram.

Architecture Structure Diagram (ASD): A SysML block diagram in the context of INTO-CPS is referred to as Architecture Structure Diagram (ASD), as defined in D2.1a [2].

An ASD is used to model SysML blocks, i.e. the external structure of the FMU along with its input/output ports and possible related interfaces, as presented in [13], and illustrated in Figure 1. An ASD can be specific to a single FMU in the overall design, and is henceforth usually used to model that single FMU in the form of a SysML block.

Figure 1 and Figure 2 respectively show two FMUs belonging to a cascading water tank example developed in

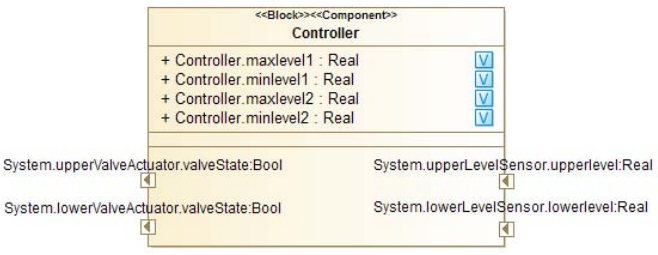


Figure 1 Example of INTO-CPS SysML Architecture Structure Diagram (Controller FMU)

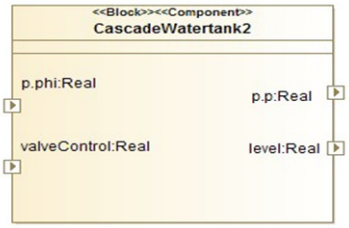


Figure 2 Cascade Water Tank FMU

20-Sim. Figure 1 illustrates the SysML block corresponding to the 'Controller' FMU and its inputs/outputs; while Figure 2 subsequently displays one of the water tank FMUs that are connected to the controller FMU.

Connection Diagram (CD): A SysML internal block diagram in the context of INTO-CPS is referred to as Connection Diagram (CD), as defined in [2]. A CD is used to define the interactions between different FMUs. The CD contains instances of FMUs (SysML Block instances) which are connected together by means of connectors to the input/output ports of the FMU instances present in the CD. The FMU instances correspond to the FMUs modelled in the different ASDs, as seen in Figures 1 and 2. The CD also enables to associate existing FMUs to the block instances for eventual simulation [13].

As seen in Figure 3, the CD illustrates the 'instance' that indicates the overall system containing FMU instances of 'Controller', 'CascadeWatertank1' and 'Cascade Watertank2'; and the instances of these FMUs are connected with each other by means of connectors to their respective input/output ports.

7 DISCUSSION AND CONCLUSION

This paper provides an overview of the current INTO-CPS SysML concepts and their relation to the FMI 2.0 standard. The paper also provides an overview on how SysML is used in INTO-CPS, by highlighting the notion of Architecture Structure and Connection Diagrams in their initial version as for the first year of the project. The authors identified that the current approach is straight forward and easy to grasp by the end-users, though it might reveal scalability problems for complex co-simulations in real industrial settings. For example, modeling all the variables as ports may be tedious for FMUs with hundreds of variables. Moreover, relating those variables explicitly with specific links may cause usability problems. Thus, for the second year of the project the authors proposed to explore an approach which groups the variables with domain related interfaces. This further development as well as the feedback from the end-users will be reported in our future papers.

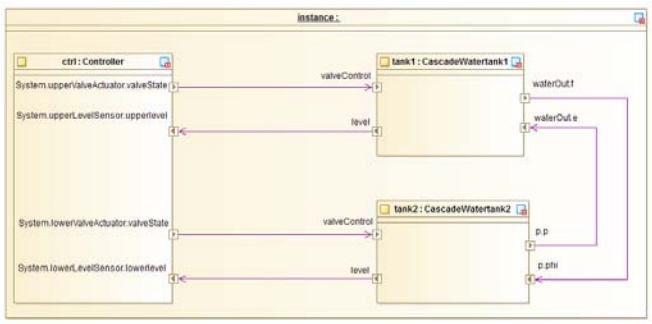


Figure 3 Example of an INTO-CPS SysML Connection Diagram

ACKNOWLEDGMENT

This research presented in this chapter is funded by the European Community's H2020 framework under grant agreement no. 644047 (INTO-CPS).

REFERENCES

[1] INTO-CPS (2015), *Deliverable 3.1a: Method Guidelines*. Available at: www.into-cps.au.dk.

[2] INTO-CPS (2015), *Deliverable 2.1a: Foundations of the SysML Profile for CPS Modeling*. Available at: www.into-cps.au.dk.

[3] FMI Standard (2016), *FMI 2.0*. Available at: https://www.fmi-standard.org/.

[4] Torsten Blochwitz Ed. (2014), *Functional mock-up interface for model exchange and co-simulation*. Available at: https://www.fmi-standard.org/downloads.

[5] Object Management Group Inc. (2003), *MOF 2.0 core final adopted specification*. Available at: http://www.omg.org/cgi-bin/doc?ptc/03-10-04.

[6] Eclipse Foundation (2016), *Eclipse Modeling Framework*. Available at: http://www.eclipse.org/emf.

[7] Object Management Group (2015), *Unified Modeling Language 2.5*. Available at: www.omg.org/spec/UML.

[8] Object Management Group (2015), *System modeling language specification v1.4*. Available at: http://www.omg.org/spec/SysML/1.4/ .

[9] Softeam (2016), *Modelio Open-Source Modeling Environment*. Available at: http://www.modelio.org/.

[10] S. Sendall and W. Kozaczynski (2003), *Model Transformation: The Heart and Soul of Model-Driven Software Development*, IEEE Software 20(5):42–45.

[11] T. Mens and P. V. Gorp (2006), *A taxonomy of model transformation*, In Proceedings of the International Workshop on Graph and Model Transformation, volume 152 of Electronic Notes in Theoretical Computer Science, pp. 125–142.

[12] 20-Sim (2016). Available at: http://www.20sim.com/

[13] INTO-CPS (2015), *Deliverable D4.1a. User Manual for the INTO-CPS Tool Chain*. Available at: www.into-cps.au.dk.

Tackling Uncertainty in Cyber-Physical Systems with Automated Testing

Shaukat Ali

Simula Research Laboratory, P.O. Box 134,1325 Lysaker, Norway; email: shaukat@simula.no

Tao Yue, Man Zhang

Simula Research Laboratory and the University of Oslo, P.O. Box 134,1325 Lysaker, Norway; email: {tao, manzhang}@simula.no

Abstract

The U-Test-EU [1] project aims at developing new methods and techniques for testing Cyber-Physical Systems (CPSs) under uncertainty. This paper aims to provide the current status of the results achieved in the project during the first one and half years. Our ultimate aim is to enable collaboration among several Horizon2020 projects focusing on CPSs. This paper focuses on the research results from the following four perspectives in the context of the project: 1) Understanding uncertainty in CPSs, 2) Modeling uncertainty in CPSs to support automated testing, 3) Discovering unspecified uncertainties, 4) Testing CPSs under the specified and discovered uncertainties. In addition to the research results, we also present a set of standardization activities that are planned in the project with the final goal of bringing results to a wider audience than the targeted projects and consortium of the project.

Keywords: Cyber-Physical Systems, Uncertainty, Model-Based Testing, Search-Based Testing.

1 Introduction

Uncertainty in Cyber-Physical Systems (CPSs) cannot be evaded and must be tackled explicitly in a systematic way starting from their development to testing and even after deployment. The current state-of-the-art and state-of-the-practice lack the systematic and automated approaches and methods to test CPSs under both *known* and *unknown* uncertainty [1; 4; 12] [3]. The U-Test-EU project aims to develop such automated and systematic approaches to test CPSs that explicitly consider uncertainty *known* at the design time in addition to discovering *unknown* uncertainties using search-based techniques such as genetic algorithms.

The aim of this paper is to present the current status of the results produced in the project to facilitate collaboration among other EU projects related to Cyber-Physical Systems. More specifically, we present the results related to 1) *Understanding* uncertainty in CPSs, 2) *Modeling known* uncertainty with the purpose of supporting automated testing, 3) *Discovering unknown* uncertainty, and 4) *Testing* CPSs under specified and discovered uncertainties. Notice that in this paper, we only present high-level details of solutions and their associated key results to facilitate discussion among projects. However, we provide appropriate references, where further details can be consulted. In addition, in the footnote[2], we provide links to the slides that were presented during the workshop for sharing the results with other projects.

The rest of the paper is organized as follows: Section 2 presents the results related to understanding uncertainty in CPSs, Section 3 discusses our modeling solution to model test ready models of CPSs, Section 4 discusses our solution for evolving test ready models to discover unknown uncertainties, and Section 5 presents the current status of testing solutions. Section 6 presents the planned standardization activities and finally we conclude the paper in Section 7.

2 Understanding Uncertainty with U-Model

In terms of understanding uncertainty in CPSs, the key outcome is an uncertainty conceptual model (U-Model) for Cyber-Physical Systems presented in [17]. Due to the lack of common understanding of uncertainty in the current literature, we developed the U-Model with the aim to provide a unified understanding of uncertainty in CPSs. In addition, we aimed to classify uncertainties in CPSs at the three logical levels of CPSs including Application, Infrastructure, and Integration levels [17]. Since there wasn't any existing uncertainty model in the context of CPSs available, we developed the U-Model by reviewing existing literature from other domains such as physics, healthcare, and statistics [17].

The U-Model took a subjective approach to understanding uncertainty in CPSs, where a belief agent (e.g., a modeler or a group of modeler) holds some belief about some aspects of CPSs (test ready models in our context to generate test cases). The U-Model has three sub-models as

[1] www.u-test.eu

[2] http://www.cister.isep.ipp.pt/ae2016/presentations/utest1.pdf
http://www.cister.isep.ipp.pt/ae2016/presentations/utest2.pdf

described in [17]: *Belief Model*, *Uncertainty Model*, and *Measure Model*. The *Belief Model* captures the basic concepts related to beliefs and belief agents, whereas the *Uncertainty* model captures the concepts specifically related to uncertainty such as various types of uncertainties, patterns, and measurements. The *Measure Model* aims at capturing uncertainty measures at a very high level. More details of the *U-Model* and its associated sub-models can be consulted in [17]. The U-Model was specialized into three uncertainty taxonomies at each level of CPSs, i.e., *Application*, *Infrastructure*, and *Integration*. More detailed taxonomies are presented in the associated technical report [17] and deliverable on the project website[3].

In order to validate the completeness and correctness of the U-Model, we validated it using uncertainty requirements collected from the two CPSs case studies that are available to us as part of the project [17]. The first case study is from the healthcare domain and is called GeoSports (GS) provided by Future Position X (FPX), Sweden (www.fpx.se). The second CPS case study is about Automated Warehouse (AW) provided by ULMA Handling Systems, Spain (http://www.ulmahandling.com/). Details of the validation are also presented in [17].

3 Modeling Test Ready Models with Uncertainty using The Uncertainty Modeling Framework

The second key result of the project is the Uncertainty Modeling Framework (UMF) presented in [13]. The main objective of the UMF is to provide a standard-based modeling framework to create test ready models of a CPS with explicit consideration of uncertainty. The test ready models created with the UMF are used by the Uncertainty Testing Framework (UTF) (Section 5) as input to generate test cases for execution based on various test strategies [13].

The core of the UMF relies on the implementation of the U-Model as a UML profile called the UML Uncertainty Profile (UUP) [13] that allows modeling concepts related to beliefs and uncertainties defined in the U-Model on UML models. In addition, the UMF uses the UML Testing Profile (UTP) V.2 [9] to make the models test ready. In addition, to facilitate modeling uncertainty with a variety of uncertainty measurements, we have created an extensive set of model libraries including *Measure*, *Pattern*, *Time* and *Risk* libraries. These libraries extend the UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE) [10]. Within the UMF we have defined a set of guidelines to model test ready models with UUP, UTP, and the model libraries at the three testing levels of CPSs including Application, Infrastructure, and Integration levels. Interested readers may consult [13] for further details on the UMF.

A preliminary evaluation of the UMF was performed using the two industrial case studies, i.e., GS and AW in addition to one open source case study available from the literature [11]. The evaluation involved creating test ready models for the three case studies. The UMF was evaluated from several aspects including 1) *Completeness* and *correctness* of the various parts of the UMF were evaluated including the UUP and its associated model libraries with respect to the U-Model and the MARTE profile. The U-Model is at the core of the UUP, whereas the MARTE profile is at the core of the model libraries, 2) Effort required to create test ready models with UMF in terms of time for the three case studies, 3) To check the correctness of the test ready models in terms of wrong model elements, incomplete model elements, and redundant model elements, the test ready models were executed with test data using the IBM Rational Software Architecture (RSA)'s simulation toolkit [5]. The detailed results are presented in the technical reports [13; 15].

4 Evolving Test Ready Models with the U-Evolve Framework

The third key result of the project is a preliminary version of the model evolution approach embedded in the U-Evolve framework [16]. The overall aim of U-Evolve is to take input the test ready models created with the UMF and evolve the models with the aim of discovering new uncertainties [16].

The U-Evolve includes several steps to evolve the test ready models; however, at its current stage, the U-Evolve uses dynamic inference techniques to discover uncertainties. Since the dynamic techniques require data, we used the real data available for the case study from its actual use. The U-Evolve works in the following three steps [16]:

In the first step, we verify the initial version of test ready models with the real data. Test ready models explicitly capture uncertainty. The verification is performed by executing the test ready models with real data by enriching the models with the UML Action Language (UAL) code, which is based on the Action Language for Foundational UML (ALF) [4] standard. The IBM RSA's simulation toolkit [5] was used for this purpose. In the second step, the objective uncertainty values were introduced to the verified test ready models based on the real data. In the third step, we used the Daikon tool [2] that applies machine learning techniques to infer likely invariants based on data. In our context, we used the Daikon tool [2] to further refine constraints on the test ready models based on the real data. The key constraints were specified in the Object Constraint Language (OCL)[5] included: State Invariants on states in UML state machines modeling test oracles, and guard conditions on transitions of UML state machines specifying test data specifications.

[3] www.u-test.eu

[4] http://www.omg.org/spec/ALF/

[5] http://www.omg.org/spec/OCL/

The U-Evolve framework was evaluated with the GS case study as part of the project. More details on the U-Evolve framework and results can be found in the technical report [16].

5 Uncertainty-based Test Case Generation and Minimization

One of the key activities in the project is to generate test cases from the test ready models developed using the UMF and also from the evolved test ready models after using U-Evolve [16]. We have performed some preliminary work in this part, where we have defined in total two test case generation and four test case minimization strategies using multi-objective search algorithms relying on uncertainty theory [6]. The test strategies are implemented in a tool called U-TCsMG and further details can be consulted in [14].

For U-TCsMG [14], first, we conducted an empirical study using the SafeHome case study to evaluate the proposed test strategies. Based on the results of the empirical evaluation, we selected the best test strategy (test case generation followed by test case minimization), which managed to minimize test cases up to 91% and achieved 100% mutation score [14]. With the best test strategy, we tested the real industrial case study of GeoSports. The results showed that from the total of 2085 test cases generated, the best strategy managed to reduce the test cases to 336 (83.9%). We executed the minimized set of test cases on the real system and managed to found 98 uncertainties (incorrect locations of devices). Of these 98 observed uncertainties, 80 were because of the intentionally introduced indeterminacy source, where 18 occurred because of the unknown reason(s). We are analyzing the observed uncertainties to find the indeterminacy sources to prevent them happening in the future executions.

6 Standardization

To achieve the wider impact of the results produced in the project, one of the key activities is to standardize some of the results of the project. As part of the project, we are working in the three directions: 1) The standardization of the UML Testing Profile V.2 in the Object Management Group (OMG). Our efforts in this direction can be followed at [9], 2) Initiation of a new standard corresponding to the U-Model and the UUP profile at the OMG. The efforts in this direction can be followed at [8] , 3) Towards the end of the project, we are planning to write recommendations based on the project results to various standardization bodies such as the European Telecommunications Standards Institute (ETSI). In this direction, we are already contributing to the development of Request For Proposals (RFP) of Systems Modeling Language (SysML) V.2 based on the results from the project. The progress can be followed at [7].

7 Conclusion

This paper presented the results achieved in the first one and half years of the U-Test-EU project. This involved the results related to understanding uncertainty, modeling uncertainty, discovering uncertainty, and testing CPSs under uncertainty in the context of the ongoing EU project. In addition, we also presented the standardization efforts that are being performed as part of the project.

Acknowledgment

This work was supported by the EU Horizon 2020 funded project (Testing Cyber-Physical Systems under Uncertainty). Tao Yue and Shaukat Ali are also supported by RCN funded Zen-Configurator project, RFF Hovedstaden funded MBE-CR project, RCN funded MBT4CPS project, and RCN funded Certus SFI.

References

[1] M. Broy (2013), *Engineering Cyber-Physical Systems: Challenges and Foundations*, Proceedings of the Third International Conference on Complex Systems Design & Management CSD&M 2012.

[2] M.D. Ernst, J. Cockrell, W.G. Griswold, D. Notkin (2001), Dynamically discovering likely program invariants to support program evolution, *IEEE Transactions on Software Engineering, 27*(2), 99-123. doi:10.1109/32.908957

[3] P.C. Evans, & M. Annunziata (2012), *Pushing the Boundaries of Minds and Machines*, Paper presented at the General Electric (GE).

[4] H.-M. Huang, T. Tidwell, C. Gill, C. Lu, X. Gao, & S. Dyke (2010), *Cyber-Physical Systems for Real-Time Hybrid Structural Testing: A Case Study*, Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems.

[5] IBM (2016), *IBM RSA Simulation Toolkit*. www-03.ibm.com/software/products/en/ratisoftarchsimutool, Retrieved on November 11, 2016.

[6] B. Liu (2015), *Uncertainty theory*, Springer.

[7] Object Management Group (2016), Requirement Concepts Modeling Focus Team http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:requirement_concepts_modeling_core_team, Retrieved on November 11, 2016.

[8] Object Management Group (2016), Uncertainty Modeling (UM) http://www.omgwiki.org/uncertainty/doku.php?id=start, Retrieved on November 11, 2016.

[9] Object Management Group (2016), UML Testing Profile, http://utp.omg.org, Retrieved on November 11, 2016

[10] Object Management Group (2016), UML Profile For MARTE: Modeling And Analysis Of Real-Time Embedded Systems, http://www.omg.org/omgmarte/, Retrieved on November 11, 2016.

[11] R.S. Pressman (2010), *Software engineering: a practitioner's approach 7th edition*, Palgrave Macmillan.

[12] T. Tidwell, X. Gao, H.-M. Huang, C. Lu, S. Dyke, & C. Gil (2009), *Towards Configurable Real-Time*

Hybrid Structural Testing: A Cyber Physical Systems Approach, Proceedings of the 2009 IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing.

[13] M. Zhang, S. Ali, T. Yue, & P. H. Nguyen (2016), *Uncertainty Modeling Framework for the Integration Level V.1*. https://www.simula.no/publications/uncertainty-modeling-framework-integration-level-v1

[14] M. Zhang, S. Ali, T. Yue, & M. Hedman (2016), *Uncertainty-based Test Case Generation and Minimization for Cyber-Physical Systems: A Multi-Objective Search-based Approach.* https://www.simula.no/publications/uncertainty-based-test-case-generation-and-minimization-cyber-physical-systems-multi

[15] M. Zhang, S. Ali, T. Yue, & R. Norgre (2016), *An Integrated Modeling Framework to Facilitate Model-Based Testing of Cyber-Physical Systems under Uncertainty*. https://www.simula.no/publications/integrated-modeling-framework-facilitate-model-based-testing-cyber-physical-systems

[16] M. Zhang, S. Ali, T. Yue, & R. Norgren (2016), *Interactively Evolving Test Ready Models with Uncertainty Developed for Testing Cyber-Physical Systems*. https://www.simula.no/publications/interactively-evolving-test-ready-models-uncertainty-developed-testing-cyber-physical

[17] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, & R. Norgren (2016), *Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model*, European Conference on Modelling Foundations and Applications (ECMFA). https://www.simula.no/publications/understanding-uncertainty-cyber-physical-systems-conceptual-model

Spatio-temporal Model-Checking for Collective Adaptive Systems in QUANTICOL

V. Ciancia, D. Latella
Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo", Pisa, Italy; email: {Diego.Latella, Vincenzo.Ciancia}@cnr.it
M. Loreti
Università di Firenze – Dipartimento di Statistica, Informatica, Applicazioni, Firenze, Italy and IMT Alti Studi, Lucca, Italy; email: Michele.Loreti@unifi.it
M. Massink
Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo", Pisa, Italy; email: Mieke.Massink@cnr.it

Abstract

Spatial aspects of computation are becoming increasingly relevant when dealing with systems distributed in physical space. Traditional formal verification techniques are well suited to analyse the temporal evolution *of system models; however, properties of* space *are typically not taken into account explicitly. In this position paper we briefly review some of the recent developments of spatial and spatio-temporal model-checking in the context of the European research project QUANTICOL funded by the FET-Proactive programme on Fundamentals of Collective Adaptive Systems. We illustrate some typical applications of spatial and spatio-temporal model checking on collective adaptive systems and provide an outline for further developments.*

Keywords: Temporal Logics, Spatial Logics, Model-checking, Collective Adaptive Systems.

1 Introduction

The concept of smart cities is on the research agenda of many EU and other international institutions and think-tanks. Although not the only factor for success of smart cities, innovative ICT-based technology is seen by many as a key factor that would allow modern cities to reach or maintain a good and sustainable quality of life for their inhabitants, with timely and equitable distribution of resources. At the core of many proposals ranging from smart buildings and transportation to a smart electricity grid, is the transformation of a centralised system architecture and control to a much more decentralised and distributed design. Similar issues of optimal distribution and congestion avoidance play a role in smart transportation, whether based on public transport or community initiatives such as shared bikes.

The very fact that such systems are highly distributed and their adaptive behaviour relies on the tight and continuous feedback between vast numbers of consumers and producers, makes such systems typical examples of large scale collective adaptive systems (CAS). These are systems that consist of a large number of *spatially distributed* heterogeneous entities with decentralised control and varying degrees of complex autonomous behaviour. QUANTICOL[1] is a research project funded by the FET-Proactive programme on Fundamentals of Collective Adaptive Systems. It aims to develop novel quantitative analysis techniques to support the design and operational management of a wide range of collective adaptive systems, with particular focus on applications arising in the context of smart cities.

Spatial aspects of computation are becoming increasingly relevant when dealing with systems distributed in physical space. Traditional formal verification techniques are well suited to analyse the temporal evolution of system models; however, properties of space are typically not taken into account explicitly. The global behaviour of CAS critically depends on interactions which are often local in nature, and thus aspects of locality immediately raise issues of spatial distribution of objects.

One of the project's proposals to facilitate *reasoning* about spatial aspects of CAS is the development of *spatial model-checking*. Model checking has been widely recognised as a powerful approach to the automatic verification of concurrent and distributed systems (see [1] and references therein). It consists of an efficient procedure that, given an abstract model M of the system, decides whether M satisfies a logical formula Φ. Traditionally, such formulas are drawn from a temporal logic and used to verify temporal aspects of a system such as "there exists a run of the system that eventually reaches a state in which the queue is full". Such temporal logics have later been extended with probabilistic and stochastic notions allowing for the verification of properties such as "the probability is 0.1 that the system reaches a state in which the queue is full" [1, 2, 3]. In the context of the QUANTICOL project stochastic and probabilistic model-checking has been further extended to address large scale CAS using model-checking techniques based on fluid and mean field approximations originating from the area of statistical physics [4, 5].

[1] Web site: `www.quanticol.eu`

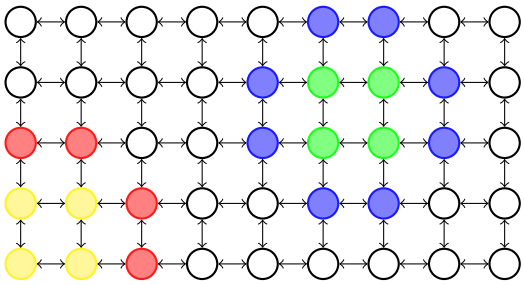


Figure 1: A graph inducing a *quasi-discrete* closure space

In spatial model-checking, instead, one is interested in verifying properties of *space*. Typical spatial properties concern questions of being near to a place satisfying a certain property, or of being reachable through space or of being surrounded by particular points. Spatial model-checking requires a spatial logic and a spatial representation on which such a logic can be interpreted, and, of course, efficient spatial model-checking algorithms. Furthermore, spatial model-checking can be combined with temporal model-checking leading to spatio-temporal model-checking. This gives rise to the verification of properties concerning the behaviour of a system in space and time. For example, in a collective system such as bike sharing one could then verify complex properties such as "eventually, when a station is full, there is a moment in the future in which all its adjacent stations will be full as well".

In this position paper we provide a brief overview of some of the recent developments on spatial model-checking in the context of the QUANTICOL project and some pointers to related publications.

2 Spatial Logic for Closure Spaces

The development of spatial logics dates back to the work by early logicians such as Tarski, who studied possible semantics of classic modal logics, using topological spaces. Topological spaces may be seen as generalisations of Euclidean spaces by focussing on the notion of *closeness* without making reference to an explicit metric. The field of spatial logics is well developed in terms of descriptive languages and aspects such as computability and complexity [6], but does not yet address formal verification problems. In particular, discrete spatial models are still a relatively unexplored field. One of the spatial logics, the Spatial Logic for Closure Spaces (SLCS) [7], proposed in the context of the QUANTICOL project is based on so-called Closure Spaces. These are a generalisation of topological spaces that include both continuous and discrete spatial models, among which the widely used discrete mathematical structure of graphs. Graphs are extremely versatile. Their use includes, for example, the representation of digital images. Graphs give rise to the subset of Closure Spaces which are known as Quasi-discrete Closure Spaces [8].

The logic SLCS builds on the tradition of modal logics and on the modal logics approach to spatial logics in which the two well-known modalities *possibility* and *necessity* are given a topological interpretation, namely that of *closure* and its dual *interior* (on the reals or a similar metric space) [6]. In SLCS both modalities have been given an interpretation suitable for reasoning about *discrete* spaces. Besides these two basic notions, a further logical operator has been introduced, namely the *surrounded* operator. This operator takes inspiration from the well-known temporal until-operator but is casted and re-interpreted in a discrete spatial setting. In summary, SLCS is equipped with two spatial operators: a "one step" modality, called "near" and denoted by $\mathcal{N}$, turning the closure operator into a logical operator, and a binary spatial until operator $\Phi_1 \mathcal{S} \Phi_2$. The basic idea is that a point x in the (quasi-discrete closure) space satisfies $\mathcal{N}\Phi$ if it is adjacent to a point that satisfies Φ. For instance, if we consider the model of Fig. 1, the green and the blue nodes satisfy $\mathcal{N}\,green$. The dual operator of $\mathcal{N}$ is the interior $\mathcal{I} = !\mathcal{N}(!\Phi)$. The green nodes satisfy $\mathcal{I}(green \cup blue)$.

A point x satisfies $\Phi_1 \mathcal{S} \Phi_2$ whenever there is "no way out" from a set of points, including x, and that each satisfy Φ_1 unless passing by a point that satisfies Φ_2. For instance, in Fig. 1, *yellow* nodes satisfy $yellow \mathcal{S}\, red$ while *green* nodes satisfy $green \mathcal{S}\, blue$.

This small set of spatial logic operators, together with the basic boolean operators such as negation and conjunction, is surprisingly expressive. For example, a number of interesting derived operators can be defined, including the well-known spatial "somewhere" and "everywhere" operators, and various forms of reachability. Moreover, an efficient model-checking algorithm has been developed for this set of operators that was first presented in [7]. In [9] the logic is extended with an additional "propagation" operator, $\mathcal{P}$, such that a point x satisfies $\Phi_1 \mathcal{P} \Phi_2$ if and only if it satisfies Φ_2 and there is a path rooted in a point satisfying Φ_1 where all other points satisfy Φ_2. This operator can be useful for describing, for instance, situations in which, a "safe" point x can be reached starting from a point where something dangerous takes place (e.g. Φ_1 could model the fact that there is a source of radiation there while Φ_2 represents shielded, safe, points in space). In the above mentioned paper, the logic has also been extended with *collective* operators, which are interpreted on *sets* of points instead of individual points. Finally the model-checking algorithms have been extended accordingly. We refer to [9] for details; a tutorial on the subject is provided in [10].

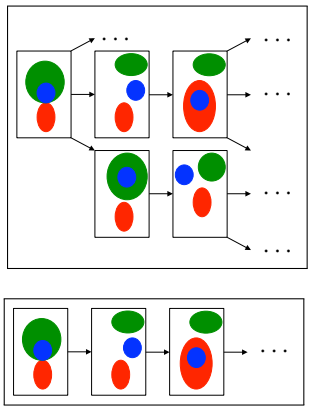


Figure 2: A temporal structure representing a computation tree of *snapshots* induced by the time-dependent valuations of the atomic propositions (top). A path in the model (bottom).

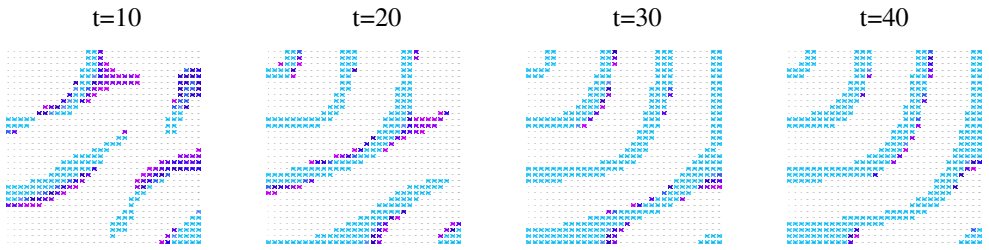


Figure 3: Formation of a wave-like pattern; evolution in steps of 10; pink points are part of a wave for 2 subsequent steps; cyan points for 10 subsequent steps. Other colours represent the intermediate number of steps.

3 Spatio-temporal Logic for Closure Spaces

The spatial logic SLCS has been extended with classical branching time temporal logic operators [11], leading to STLCS, in such a way that spatial and temporal operators can be arbitrarily nested. This way interesting properties of the spatio-temporal dynamics of a system can be expressed. In STLCS a temporal structure represents a computation tree of spatial *snapshots* induced by the time-dependent valuations of the atomic propositions (see Fig. 2). A spatio-temporal model checker was developed for STLCS and introduced in [11], called `topochecker`. This model checker[2] can verify multiple properties simultaneously and show their results in different selected colours. In case the results involve the same points, the later results overwrite the previous ones. The time complexity of the spatial model checking algorithm is linear in the number of points and arcs in the space and in the size of the formula.

To illustrate spatio-temporal model checking, we consider an example of the formation of a wave-like pattern as described in [10]. Such patterns can emerge when two particular chemical substances, or morphogens, A and B interact and diffuse over a surface in a way similar to that of the formation of Turing patterns, which were first studied and discovered by Alan Turing in his groundbreaking paper on morphogenesis in 1952 [12].

The spatio-temporal model consists of a sequence of snapshots of the first 100 time steps. This model is obtained as the numerical solution of the set of reaction-diffusion equations that describe the dynamics of the concentrations of both substances in a regular grid of discrete patches (see also [10]). The last snapshot of the sequence is repeated in an artificial way to obtain an infinite path. Each snapshot consists of a regular graph of 31 by 31 discrete patches, where each node is connected to its four direct neighbours. Each patch represents the local concentration of the two substances.

The spatio-temporal logic can be used to identify which points (denoting patches) are part of a pattern for a number of consecutive steps in the dynamic evolution of the space. First we define the spatial property 'pattern' as an area of points with low concentration `a` of chemical substance A surrounded by points with higher concentration of A:

$$\texttt{pattern} \quad = [\texttt{a<0}]\texttt{S}[\texttt{a>=0}]$$

[2] Available at http://www.github.com/vincenzoml/topochecker.

Then we define the various periods, ranging from 3 to 10 time steps, during which a point remains part of the pattern as follows (using the front end notation of `topochecker` for STLCS formulas):

$$\begin{aligned}
&\texttt{pattern2steps} &&= \texttt{pattern} \,\&\, \texttt{AX}(\texttt{AX}\,\texttt{pattern}) \\
&\texttt{pattern3steps} &&= \texttt{pattern2steps} \,\&\, \texttt{AX}(\texttt{AX}(\texttt{AX}\,\texttt{pattern})) \\
&\cdots \\
&\texttt{pattern10steps} &&= \cdots
\end{aligned}$$

Here operator `A` denotes 'for all paths' and `X` is the next step operator from temporal logics. So the formula `pattern2steps` is satisfied by points that satisfy property 'pattern' now, and for all paths (there is only one in this single linear sequence) in the next snapshot the point satisfies 'pattern' in the next snapshot (on all paths). We can verify such properties starting from the initial snapshot, but also starting from any other chosen snapshot in the sequence.

Figure 3 shows the evolution of the wave-like pattern when the formulas are evaluated taking as initial snapshot the one at time 10, 20, 30 and 40, respectively. The results show that the pattern seems to stabilise starting from the north-western corner of the figure after which the points towards the south-eastern corner become increasingly stable, at least for 10 subsequent steps in time.

The spatial logic can be applied on any finite graph structure. The results for a 3D version of the wave-like pattern is shown in Fig. 4. The colours in that figure have the only purpose of being able to distinguish the pattern in a 3D representation. All coloured points satisfy the property "pattern" introduced before.

Another example shows how STLCS can be used to detect the formation of clusters of full bike stations in a simulation of a model of a bike sharing system [13, 14]. The bike sharing model has a number of stations comparable to that of a city like London, but for simplicity they are assumed to be placed on a regular graph of 19 by 38 nodes (see Fig. 5). Full stations and clusters of full stations can be defined as:

$$\begin{aligned}
&\texttt{full} &&= [\texttt{vacant==0}] \\
&\texttt{cluster} &&= \texttt{I}(\texttt{full})
\end{aligned}$$

Here $[\texttt{vacant==0}]$ is an atomic proposition and `I` denotes the *interior* of a set of points (nodes). A point denoting a station evolves into a cluster when it becomes full, and stays full until it becomes part of a cluster. This may be detected

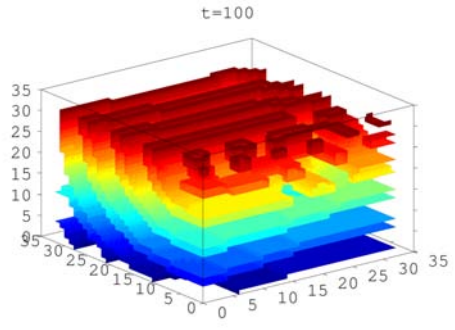


Figure 4: Analysis of property "pattern" in a 3D structure.

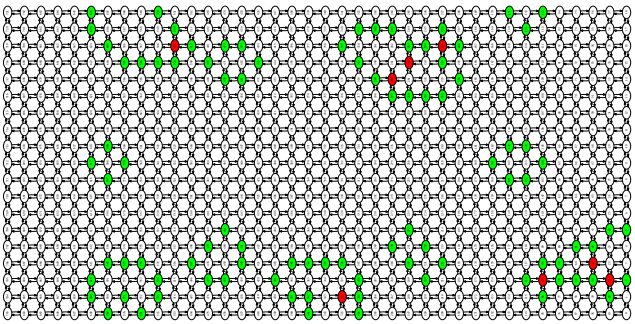


Figure 5: Formation of clusters (red) and boundary of points that will become a cluster (green).

by using the following formulas and using different colours to visualise the model-checking results as shown in Fig. 5:

```
implies(f,g)   = (!f)|g;
nextCluster    = (EF full)&
                   (AG implies(full,
                     A full U cluster))
```

The definition of `nextCluster` characterises points that will eventually become full and, for every future state, whenever full, they will remain full until they become part of a cluster. Such points are central in the formation of clusters, as they represent stations that always form a cluster when they become full. In Fig. 5, these points are shown in red, in a state of the simulation where there are many of them. For comparison, the boundary of the points that will become a cluster are shown in green, that is, those points satisfying `(NEF cluster)&(!EF cluster)`.

These are only a few examples that illustrate the use and potential of spatial and spatio-temporal model-checking in the context of CAS. Further details and examples can be found in a recent tutorial by the authors on spatial logic and spatial model-checking for closure spaces [10] and the references therein.

4 Conclusions and Outlook

We have provided a brief overview of recent work on the development of spatial and spatio-temporal model-checking for the analysis of Collective Adaptive Systems in the context of the EU FET-Proactive project QUANTICOL. The operators of the spatial logic SLCS have been inspired by topological operators and by a spatial version of the until-operator of temporal logic.

A prototype proof-of-concept spatio-temporal model-checker `topochecker` has been developed and used for the analysis of the dynamic spatio-temporal behaviour of various collective adaptive systems. Operators of the spatial logic have also been combined with a temporal logic for signals in [15] and provided with a quantitative semantics to assess the robustness with which formulas are satisfied.

Other recent work extends the approach to spatio-temporal *statistical* model-checking based on a statistical analysis of sets of simulations [16]. This approach provides insight in the *probability* with which a spatio-temporal property holds. For example, one can assess the probability that stations in a bike sharing system will get full. The approach exploits the use of a *single* set of simulations for the spatio-temporal properties of *all* points in a quasi-discrete closure space by means of the MultiVeStA [17] tool combining `topochecker` and the simulator for bike sharing models [13].

Future work is planned on the extension of the approach with suitable metric spaces and further operators, in particular for the application of the approach in the domain of medical imaging. Preliminary work on these ideas can be found in [18]. Furthermore, a possible integration of spatial model checking with highly scalable mean-field based model checking is envisioned and the exploration of suitable spatial model reduction methods. We also plan to investigate the development of a spatio-temporal model checker as a suitable combination of `topochecker` and an ADA implemented temporal model checker of the KandISTI family [19, 20].

Acknowledgements This work is supported by the EU project *QUANTICOL: A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours*, 600708. The authors would like to thank Luca Bortolussi, Stephen Gilmore, Gianluca Grilletti, Laura Nenzi, Rytis Paškauskas and Andrea Vandin who are involved in the QUANTICOL project and who are co-authors of the various articles involving spatial and spatio-temporal model-checking closely related to the work mentioned in the present position paper.

References

[1] C. Baier and J.-P. Katoen (2008), *Principles of model checking*, MIT Press.

[2] C. Baier, B. R. Haverkort, H. Hermanns, and J. Katoen (2003), *Model-checking algorithms for continuous-time markov chains*, IEEE Trans. Software Eng., vol. 29, no. 6, pp. 524–541.

[3] M. Z. Kwiatkowska, G. Norman, and D. Parker (2011), *PRISM 4.0: Verification of probabilistic real-time systems*, Computer Aided Verification - 23rd International Conference Proceedings, (G. Gopalakrishnan and S. Qadeer, eds.), vol. 6806 of LNCS, pp. 585–591, Springer.

[4] L. Bortolussi and J. Hillston (2015), *Model checking single agent behaviours by fluid approximation*, Inf. Comput., vol. 242, pp. 183–226.

[5] D. Latella, M. Loreti, and M. Massink (2015), *On-the-fly PCTL fast mean-field approximated model-checking for self-organising coordination*, Sci. Comput. Program., vol. 110, pp. 23–50.

[6] J. van Benthem and G. Bezhanishvili (2007), *Modal logics of space*, Handbook of Spatial Logics, pp. 217–298, Springer.

[7] V. Ciancia, D. Latella, M. Loreti, and M. Massink (2014), *Specifying and Verifying Properties of Space*, Theoretical Computer Science - 8th IFIP International Conference, J. Díaz, I. Lanese, and D. Sangiorgi, eds.), vol. 8705 of LNCS, pp. 222–235, Springer.

[8] A. Galton (2003), *A generalized topological view of motion in discrete space*, Theoretical Computer Science, vol. 305, no. 1–3, pp. 111 – 134.

[9] V. Ciancia, D. Latella, M. Loreti, and M. Massink (2016), *Model Checking Spatial Logics for Closure Spaces*, Logical Methods in Computer Science, vol. 12, Issue 4.

[10] V. Ciancia, D. Latella, M. Loreti, and M. Massink (2016), *Spatial logic and spatial model checking for closure spaces*, 16th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Quantitative Evaluation of Collective Adaptive System, (M. Bernardo et al., ed.), vol. 9700 of LNCS, pp. 156–201, Springer.

[11] V. Ciancia, G. Grilletti, D. Latella, M. Loreti, and M. Massink (2015), *An experimental spatio-temporal model checker*, Software Engineering and Formal Methods - Collocated Workshops, Revised Selected Papers, vol. 9509 of LNCS, pp. 297–311, Springer.

[12] A. M. Turing (1952), *The Chemical Basis of Morphogenesis*, Philosophical Transactions of the Royal Society of London B: Biological Sciences.

[13] M. Massink and R. Paškauskas (2015), *Model-based assessment of aspects of user-satisfaction in bicycle sharing systems*, Proceedings of the 18th IEEE International Conference on Intelligent Transportation Systems, (Sotelo Vazquez, M. et al., ed.), pp. 1363 –1370, IEEE.

[14] V. Ciancia, D. Latella, M. Massink, and R. Pakauskas (2015),*Exploring spatio-temporal properties of bike-sharing systems*, IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp. 74–79, IEEE Computer Society.

[15] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink (2015), *Qualitative and quantitative monitoring of spatio-temporal properties*, RV, vol. 9333 of LNCS, pp. 21–37, Springer.

[16] V. Ciancia, D. Latella, M. Massink, R. Paškauskas, and A. Vandin (2016), *A tool-chain for statistical spatio-temporal model checking of bike sharing systems*, Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th International Symposium, Proceedings, Part I (T. Margaria and B. Steffen, eds.), vol. 9952 of LNCS, pp. 657–673, Springer.

[17] S. Sebastio and A. Vandin (2013), *MultiVeStA: Statistical Model Checking for Discrete Event Simulators*, ValueTools, pp. 310–315, ACM.

[18] G. Belmonte, V. Ciancia, D. Latella, and M. Massink (2016), *From collective adaptive systems to human centric computation and back: Spatial model checking for medical imaging*, Proceedings of the Workshop on FORmal methods for the quantitative Evaluation of Collective Adaptive SysTems, (M.-H. ter Beek and M. Loreti, eds.), vol. 217 of EPTCS, pp. 81–92.

[19] S. Gnesi and F. Mazzanti (2011), *An abstract, on the fly framework for the verification of service-oriented systems*, Rigorous Software Engineering for Service-Oriented Systems - Results of the SENSORIA Project on Software Engineering for Service-Oriented Computing, (M. Wirsing and M. M. Hölzl, eds.), vol. 6582 of LNCS, pp. 390–407, Springer, 2011.

[20] M. H. ter Beek, S. Gnesi, and F. Mazzanti (2015), *From EU projects to a family of model checkers - from kandinsky to kandisti*, Software, Services, and Systems - Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering, (R. D. Nicola and R. Hennicker, eds.), vol. 8950 of LNCS, pp. 312–328, Springer, 2015.

Modeling Multi-board Communication in the AXIOM Cyber-Physical System

Roberto Giorgi, Somnath Mazumdar, Stefano Viola
University of Siena, Siena, Italy; email: {surname}@dii.unisi.it
Paolo Gai, Stefano Garzarella, Bruno Morelli
Evidence S.r.l., Pisa, Italy; email: {pj, s.garzarella, b.morelli}@evidence.eu.com
Dionisios Pnevmatikatos, Dimitris Theodoropoulos
Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH) - Crete, Greece; email: {pnevmati,dtheodor}@ics.forth.gr
Carlos Alvarez, Eduard Ayguadé, Javier Bueno, Antonio Filgueras, Daniel Jimenez-Gonzalez, Xavier Martorell
Barcelona Supercomputing Center and Universitat Politècnica de Catalunya, Barcelona, Spain; email: {name.surname}@bsc.es

Abstract

The main goal of the AXIOM project is to design a small board that could be used as a LEGO^{TM}-style module to build systems with more performance while keeping the programming task simple by using a familiar shared-memory programming model. The interconnection plays a crucial role both for the need of providing fast and reliable communication (including lossless control flow as, e.g., Infiniband, but with a simplified scope and cost). In this paper, we outline some of our initial choices and explore the performance of RDMA based mechanisms and interfaces, including the remote memory management behind the programming model. Our initial results show a potential for scaling the system as we use DF-Threads, good bandwidth for RDMA transfers, promising to scale once we use the OmpSs, programming model.

1 Introduction

Currently, Cyber-Physical Systems (CPSs) are not only a part of our personal life but also show their importance in industry or the so-called *Industry 4.0* [1]. To deploy more computational power while keeping the system modular and easy to program, we are exploring faster but inexpensive interconnects. Some example applications that need real-time processing are surveillance for security, improving efficiency and safety in homes, or systematic analysis of the interactions between the engineering systems, humans, and the physical world. These expectations impose scientific and technological challenges that AXIOM (Agile, eXtensible, fast I/O Module) project [2,3,4,5,6] is exploring through new hardware as well as software architectures for CPSs in order to provide:

- Easy programming through standard software tool chain.
- Customizable logic (i.e., FPGAs).
- Extensibility (i.e., scalability).
- Easy integration for other components (such as sensors).

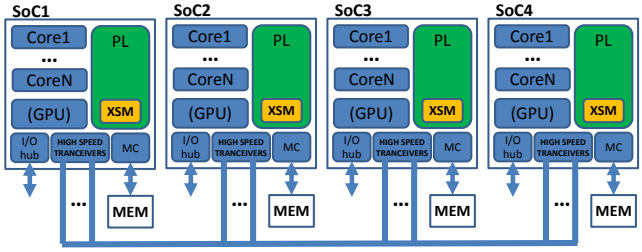


Figure 1: AXIOM Scalable Architecture. An instance consisting of four boards, each one based on the same System-on-Chip (SoC). GPU is an optional component, MC=memory controller, PL=programmable logic, XSM=eXtended shared memory

AXIOM research mainly targets to couple power-efficient multiple cores, (such as ARM core) and FPGA-based accelerators (such as Xilinx Zynq [7]), and produce prototypes of single-board computers. AXIOM incorporates a high-speed, inexpensive, board-to-board interconnects and controllers for commodity CPS peripherals. Figure 1 shows an instance of our platform including four general-purpose cores, FPGA logics and a few peripherals and sensors connected to it.

To summarise, our main contributions in this paper are:

- We motivate the need of high speed interconnect (Sec. 2).
- We detail some of the interfaces (Sec. 3).
- We presents our initial results (Sec. 4).

2 Importance of the communication between boards

2.1 Programming Model

2.1.1 Overview

The OmpSs programming model [8] consists of a series of directives to express parallelism and the use of heterogeneous resources. Parallelism is expressed in the form of tasks. Applications are written with OmpSs incorporate data directionality

```
  // Tasks target the SMP cores available in the
  // Cluster environment
  // All dependence data is copied to the remote
  // node, if necessary
  // Tasks are scheduled following their data
  // dependences
#pragma omp target device(smp) copy_deps
#pragma omp task in(a[0:BS*BS-1], b[0:BS*BS-1]) \
                 inout(c[0:BS*BS-1])
void matrix_multiply(int BS, float a[BS][BS],
                     float b[BS][BS],
                     float c[BS][BS]) {
    for (int ia = 0; ia < BS; ++ia)
        for (int ib = 0; ib < BS; ++ib) {
            float sum = 0;
            for (int id = 0; id < BS; ++id)
                sum += a[ia][id] * b[id][ib];
            c[ia][ib] = sum;
        }
}
...
int main(int argc, char * argv[]){
  int BS = ...
  ...
    for (i=0; i < NB; i++){
      for (j=0; j < NB; j++){
        for (k=0; k < NB; k++){
          // Each matrix_multiply invocation
          // spawns a new task
          matrix_multiply(BS,A[i][k], B[k][j], C[i][j]);
        }
      }
    }
  // wait for all the tasks to be finished
  #pragma omp taskwait
  ...
}
```

Figure 2: OmpSs Directives On Matrix Multiplication

annotations on the tasks, to allow the runtime system to compute task dependences, and schedule the tasks appropriately following them. When using OmpSs with heterogeneous resources or in cluster environments, we use the same type of data directionality annotations to allow the runtime system to transfer data to/from the heterogeneous devices alternatively, other nodes in the cluster.

The typical application written in OmpSs for the cluster environment is presented in Figure 2. Observe that it is the responsibility of the lower system level runtime to support cluster, and from the point of view of the application, tasks are targeted to the SMP device. There are no changes in the source code between the execution on the SMP or the cluster environment.

2.1.2 Remote memory management

Memory management in OmpSs@cluster [9] is implemented by two subsystems: the data directory, and the data cache. The data directory and the data cache share the responsibility of knowing where the program data is located during the execution, and more specifically which is (are) the valid copy(ies). The directory keeps track of which data is in which node. The data cache manages the data that has to be transferred in and out of an individual node. Given a memory reference, the data cache in each node determines if a valid copy of such data resides on the node. Whenever a task is about to be executed on a remote node, if the data is not present on that node, the data cache will issue the input/output operations to transfer the missing data.

The cache also manages the space available on each node. Initially, a configurable amount of memory is allocated in each node. This memory area is managed by the cache subsystem to allocate and de-allocate blocks of memory to receive data from other nodes. This subsystem also takes care of keeping track of the memory address translation before executing tasks, so that the original data addresses are translated to the node-local addresses that have been allocated in the cache. The behavior of the data cache can also be configured by the user, selecting between the usual cache modes: write-through, write-back, and no-cache. The latter causes all data to be transferred in all situations, thus providing no cache services, and it is mainly useful for debugging purposes.

2.2 Transport Level

Each AXIOM board has a set of connectors (2 to 4) which can be used to create point to point connections between boards. It allows, in general, the creation of various network topologies, including ring and 2D mesh. However, there is the possibility that the user will create a “non-standard” topology (which is neither a ring nor a 2D mesh): for these topologies, we developed an adaptive an algorithm which can first discover the network topology and then to compute the routes between the node.

2.2.1 Discovery Algorithm

The initial status of the discovery node is that all nodes are connected with bi-directional point-to-point connections, forming an unknown topology. Nodes ID has not been assigned yet. Each node has some interfaces, numbered starting from 0. The user connects with node by standard means (e.g., console, wifi) to a first node, which will be called “master”. From that node, The discovery algorithm is started.

First, the master node will receive the first available node ID, which is ID 0. The node then initiates a recursive algorithm (Figure 3) on all its interfaces, which works as follows:

- If the neighbor node on the other side of a given interface already has an ID, skip it and go to the next interface.
- If the neighbor node on the other side of a given interface does not have an ID, assign it the next available ID, then start the discovery algorithm recursively on it.
- When all interfaces have been processed, return the next available ID to the calling node.

At the end of its execution, the discovery algorithm will assign the node IDs to all nodes of the network, and also will produce a table representing the topology of the network. Each row of the table represents a point to point connection, which contains a pair of tuples (ID, interface) representing both endpoints of the connection.

```
/* AXIOM Recursive Discovery Algorithm */
int ax_discvery(node, next_id)
{
  node.my_id = next_id++;
  for <each neighbour> {
    if <neighbour node already have an ID> {
      <skip it>
    } else {
      next_id = ax_discovery(neighbour, next_id);
    }
  }
  return next_id;
}

/* start the discovery algorithm on the master node */
next_id = 0;
ax_discovery(master, next_id);
```

Figure 3: Discovery Algorithm

2.2.2 Routing Table Computation

The table which is produced as a result of the discovery algorithm is then used by the routing table computation phase. In particular, the system computes, for each node, a local routing table for each node. The local routing table contains, for each node in the system, the interface to which a packet directed to that node should be forwarded. The algorithm used is based on the shortest path routing and provides only one possible forwarding interface for each node.

2.2.3 Network Packet Structure

When designing the AXIOM interface, we tried to design the packet the format in an efficient way. For that reason, we considered the following scenarios:

- A node must be able to send a control message to a neighbor node (that is, attached directly to one of the interfaces on the board), to implement the discovery algorithm.
- A node must be able to send a control message to another node in the network, to implement distributed algorithms in an efficient way.
- A node must be able to send a datagram with a size similar to an Ethernet frame to another node, to eventually allow implementing a TCP/IP connection over the AXIOM-link.
- A node must be able to request a remote DMA operation of a long data segment.

To support these scenarios, we divided the data packets between two families: Small and Long packets. The small packets are used to send control messages with a payload up to 128 bytes. For efficiency reasons, their payload is hosted inside the internal memory of the network interface. For packets that require more data like Remote DMA (RDMA) alternatively, Ethernet (TCP/IP) frames, the long packets are suitable for the purpose.

Therefore, the AXIOM Network Interface supports four types of messages to handle the different packet lengths required.

- RAW DATA: Small packet to generic node in the network;
- RAW NEIGHBOUR: Small packet to neighbor node;
- LONG DATA: Long packet to generic node in the network;
- RDMA: Remote DMA transfer between two nodes in the network;

2.3 AXIOM Interconnects

As mentioned before, AXIOM-based systems will target "mid-range" high-performance computing by utilizing reconfigurable SoCs, interconnected using high-speed serial transceivers. The communication layer is the backbone of any distributed system, and if not carefully designed and implemented can lead to network congestion, low throughput, high latency and poor overall system performance. For this reason, the AXIOM interconnect is designed with two key parameters in mind, (a) high-performance to meet application demands, and (b) low resource utilization so that it can be implemented even to small-ranged reconfigurable SoCs.

As we describe in the next section, the AXIOM interconnect provides a network interface (NI) that supports remote direct memory access (RDMA) transactions (reads and writes), as well as direct transmission of raw data and control messages. At the same time, it is based on a low operating system (OS) intrusive approach using a minimal set of interrupts, hence introducing minimal performance overheads.

3 Specification of Proposed Interfaces

3.1 Programming Model

The OmpSs Mercurium compiler [10] translates the directive annotations on the source code to calls to our Nanos++ runtime system. Tasks are created and submitted for checking their dependencies, and they are potentially inserted on the task graph if pending dependences are found. After all, dependencies have been resolved; tasks are moved to the ready queue. The scheduling policy decides in which node the task will be executed. We have a specific scheduling policy to execute tasks in the node with more data available for the execution of the task. At this point, any data transfers needed are issued before launching the task for execution.

The runtime system uses the GASNet interface [11] to transfer data in and out from remote nodes. For every block of data, the remote node is requested to allocate a buffer in the data cache (see Section 2.1), data is transferred, and then the task descriptor is sent, with the proper address translation on each of the data arguments, to point properly into the memory allocated to the remote node. From the perspective of the local node, the task is executed asynchronously. When finished, the remote node notifies about the work done to the local node, which continues keeping track of the dependencies now resolved by the finalizing task, and potentially executing successor tasks.

The current implementation uses MPI messages across nodes to implement this communication protocol. We are experimenting an interconnection infrastructure through a newly implemented GASNet conduit.

3.2 Transport Level

As noted previously, the network interface supports Small and Long packets. That two kind of packets has been carefully designed in a way to share most of the internal implementation, thus optimizing the utilization of the FPGA and the OS network stack.

In particular, the types of message described in the previous section can be mapped into two kinds of packet descriptors:

- RAW descriptor (for RAW DATA and RAW NEIGHBOUR);
- LONG descriptor (for RDMA and LONG DATA).

The RAW descriptor is used for the small packet format. Its payload (up to 128 bytes), for efficiency reasons, is embedded in the descriptor itself. On the other hand, the LONG descriptor is used for the long messages and RDMA transfer. In this case, the descriptor contains the addresses of the buffers used for the transfer. For the RDMA requests, both source, and destination addresses are provided, whereas for the LONG DATA messages, only the source address is provided, whereas the remote address is assigned by the network interface using a pool of pre-allocated buffers.

3.2.1 User-Space Networking

The AXIOM Network Interface (AXIOM NIC) is designed to handle all the communication in the user-space to avoid whenever possible the usage of system calls. This approach has been demonstrated to increase performance, reducing the CPU utilization, as it happens for example in the netmap framework [12], which uses this technique in conjunction with batch transmission.

3.2.2 Overhead Optimization

The AXIOM NIC is also developed to reduce the interrupt overhead. In particular, the network interface does not use the classical approach that notifies the CPU with an interrupt for every packet sent or received. Many interrupts, in fact, can decrease the performance of the entire system.

In contrast, the actual implementation sends an interrupt only when the status of the queue is changed:

- for RX queues: when the status passes from Empty to Not empty.
- for TX queues: when the status passes from Full to Not full.

3.3 AXIOM at Interconnects Level

As mentioned in the previous section, the AXIOM interconnect supports demanding applications with high throughput and at the same time targets processing platforms that are bundled with minimal reconfigurable resources.

High-Performance Interconnect: The AXIOM NI will support (a) RDMA read and write transactions, to enhance the performance of applications that require large memory regions, and (b) raw data and control messages exchange for minimal network control overheads. At the same time, it will

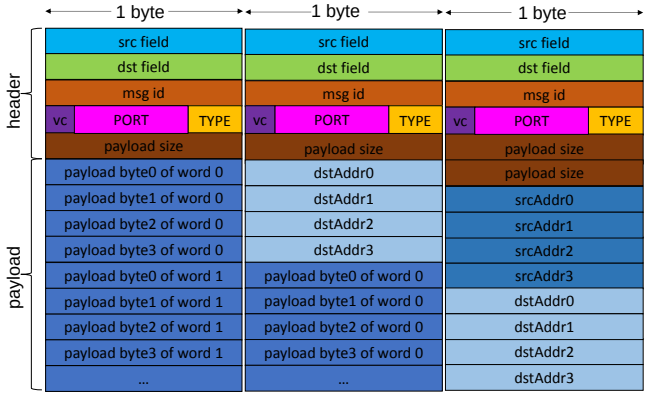


Figure 4: Packet format that support (a) raw data transmission, (b) RDMA writes and (c) RDMA reads.

follow a minimal CPU intervention approach, as it supports a minimal set of interrupts to signal an HW state change (e.g. a physical link is down), and also directly manages all local and remote ACKs. On the implementation perspective, to provide a high-performance interconnect module among all processing nodes, the AXIOM NI will provide on-chip message queues, where the OS posts message descriptors and gets updates of finished transactions, such as RDMA reads and writes.

Towards a fast and efficient network routing scheme, the AXIOM interconnect layer supports three virtual circuits (VCs); the lowest VC is used for data reads, the middle VC for data transmissions and finally the highest VC for ACKs. The physical interconnect utilizes the available serial transceivers (MGTs), wrapped by the Xilinx Aurora protocol that introduces only a 3% overhead, allowing a bandwidth of 16.375 Gbits/sec per lane [13].

The network underlying packet structure is depicted in Figure 4. It consists of a 4-byte header that provides the source/destination node ids, a unique message id (associated to software descriptor), the 2-bit routing VC, a 3-bit PORT field for implementing OS-related transactions, and finally a 3-bit TYPE field that is used to decode the packet payload. The first payload byte reports the actual payload data size (counted in words). If the packet is transmitting raw data, then all remaining bytes (up to 128) are the actual payload. However, if the packet is sending data from the local to remote memory, then its first 4 bytes designate the remote destination write address, followed by up to 128 bytes of payload. Finally, when the OS performs an RDMA read, then the first two payload bytes designate the number of requested words, followed by the remote read address and the local write address.

Low Resource Requirements: Due to the simple packet format, the AXIOM NI can efficiently process and serve posted message descriptors by the OS for transmission, hence requiring small on-chip descriptor queues. The packet router is implemented in each node's reconfigurable logic, thus not requiring additional external routing chips. Moreover, the routing logic will employ a simple 3 stage pipeline (thus fewer registers) for input buffering, route calculation - VC allocation

- switch allocation, and link traversal. The AXIOM network is based on an XY routing scheme, ensuring no deadlocks and low resource utilization. Finally, VC queue status is based on a Xon/Xoff (i.e. no credits) approach, leading to less low-level network overheads due to sync transactions.

3.4 DF-Threads: Machine Level Interface

Dynamic dataflow model [14, 15, 16, 17] supports higher parallelism and redundant execution by supporting repetitions [18, 19]. In the dataflow execution model, a program is represented by a directed graph [20]. DF-Threads [21, 22] or dataflow threads satisfy the following properties: Inheriting dataflow execution model, DF-Thread triggers its execution only when all input data are available in the frame memory (DF frame) of the producer threads. It also supports non-preemptive execution. Thread granularity is controlled by the compiler. DF threads support shared memory model also called as DF-Thread Memory Model (DFTMM) and also supports four communication models. They are 1-to-1, 1-to-N, N-to-1, and N-to-N communication. DF threads support graceful degradation [23, 24]. In the presence of failure, DF threads can continue its execution without the inherent support of checkpointing/restart technique. DF-Threads can have multiple variants (such as [25]). We have implemented six DF-Thread low-level API corresponds to C language. They are:

- `void *DF_TSCHEDULE(bool cnd, void *ip, uint64_t sc)`: Allocates the resources (a DF-frame of size `sc` words and a corresponding entry in the distributed thread scheduler or DTS) for a new DF-Thread and it returns a frame pointer `fp`. The `ip` is the instruction pointer of DF-Thread. The allocated DF-Thread is not executed until its `sc` reaches `0` and together also satisfy the boolean condition `cnd`.
- `void DF_DESTROY()`: To release allocated resources held by current DF-Thread.
- `uint64_t DF_TREAD(uint64_t offset)`: Loads the data indexed by `offset` from the current thread of DF-frame.
- `void DF_TWRITE(uint64_t val, void *fp,uint64_t off)`: The data `val` is stored into the DF-frame pointed to by `fp` at the specified offset `off`.
- `void *DF_TALLOC(uint64_t size, uint_8 type)`: Allocates a block of memory of `size` words and returns the pointer (or null) while `type` specifies the special purpose memory type.
- `void DF_TFREE(void *p)`: Frees memory pointed to by `p`.

Table 1: Message statistics

	Msg. size (bytes)	Avg. messages/s
Matmul	536 - 22000 - 32500	68 - 74
N-body	3072 - 5400 - 8192	62 - 107

Table 2: Application performance on 1 and 2 nodes (Gflops)

App / cores per node	1	2	3	4
Matmul (1 node)	0.28	0.57	0.84	1.11
Matmul (2 nodes)	0.52	1.01	1.52	1.54
N-body (1 node)	0.15	0.30	0.46	0.58
N-body (2 nodes)	0.17	0.35	0.61	0.72

4 Results

4.1 Programming Model Level

4.1.1 Message Size and Frequency

We have executed two benchmarks, matrix multiply, and N-body, in a two-node UDOO cluster, with four threads per node. We have used Extrae [26] and Paraver [27] to observe the message sizes generated by the GASNet infrastructure, and the number of messages per second send across nodes. Table 1 shows these results.

Transfers in the Matmul benchmark are larger because the matrix is divided into blocks of 128x128 single precision numbers (65536 bytes), which are split into 2 GASNet messages of around 32500 bytes. Instead, N-body deals with two basic message sizes. The block size used is 128 particles. In N-body problem particles communicate among each other. This results in data transfers of 8192 bytes for particle arrays, and data transfers of 3072 bytes for force data arrays. Regarding the average number of messages per second, both benchmarks result in similar numbers, with N-body having a little variability from 62 to 107 messages per second.

4.1.2 Overall performance

Table 2 shows the performance results obtained from the execution of the matrix multiplication and N-body benchmarks, when using 1 and 2 UDOO nodes. The results are presented in Gflops obtained.

Results on a single node show that both benchmarks scale well from 1 to Four cores. Results on two nodes show that the execution of the benchmarks scales well when going to 2 and three cores per node. Instead, scalability is limited to four cores, because the additional communication thread used by the current implementation of the OmpSs@cluster infrastructure limits the scalability. We will work to reduce the impact that this thread has in the overall performance.

It can also be observed that, while matrix multiply scales well when moving from a single node - single core to 2 nodes - single core, this is not the case for N-body. The latter obtains 0.15 Gflops in 1 node - 1 core/node and it only increases to 0.17 Gflops on 2 nodes - 1 core/node. The reason is the higher computation to communication ratio exposed by the Matmul benchmark.

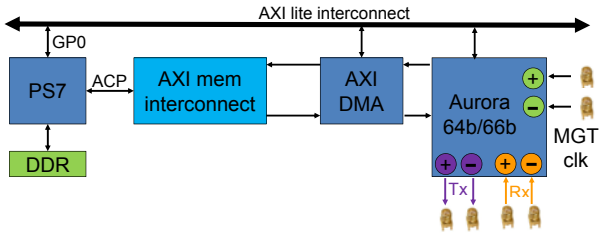


Figure 5: Block Diagram Of The Experimental System Within Each FPGA Chip

Table 3: Resource utilization of the current experimental system on the Zynq 7045 SoC.

LUTs	LUTs (%)	FFs	FFs (%)	BRAMs	BRAMs (%)
10583	4.8%	12452	2.8%	5	0.9%

4.2 Transport Level

An initial implementation of the AXIOM NIC driver has been performed on top of a set of virtual machines based on QEMU ARM 64bit, running a Linux buildroot minimal distribution. The QEMU installation has been based on top of the QEMU distribution provided for Xilinx Ultrascale+.

The architecture implemented is composed of the following components:

- A set of registers and interrupt generation routines implemented inside a QEMU frontend, used to emulate the main features of the interconnect.
- A QEMU backend that is responsible for receiving and forwarding the packets handled by the simulated over a set of host TCP/IP connections directed to a separate process simulating the network topology and routing. The network topology can be set as basic ring/2D mesh topologies, as well as custom topologies using an external description file.
- A Linux driver implementing RAW DATA and RAW NEIGHBOUR message send and receive.
- A user space library making available the AXIOM message API to user space applications.
- A set of user space applications that provide an implementation of the discovery and routing algorithms described in the previous sections, as well as a set of utility applications such as ping, traceroute, netperf, and others.

Future version of the runtime implementation will include support for RDMA and LONG DATA messages, and will provide an Ethernet frame encapsulation to give the possibility of routing TCP/IP over the AXIOM interconnect.

4.3 Interconnects Level

In our initial set of experiments, we performed a first evaluation of the DMA engine, and the physical network interconnects medium; we connected the processing system (PS7) to a DMA engine, and the latter using the AXI stream protocol with the Aurora IP. In our current configuration, the Aurora IP utilizes only one MGT transceiver. Figure 5 shows the block diagram of the experimental system that was mapped on two

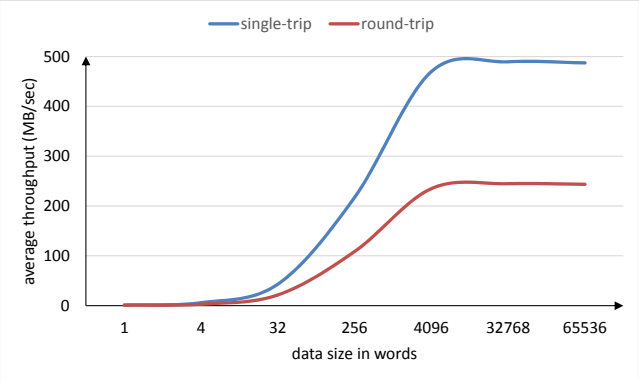


Figure 6: Network throughput when using the Aurora IP to exchange data between two ZC706 FPGA boards with a single MGT transceiver.

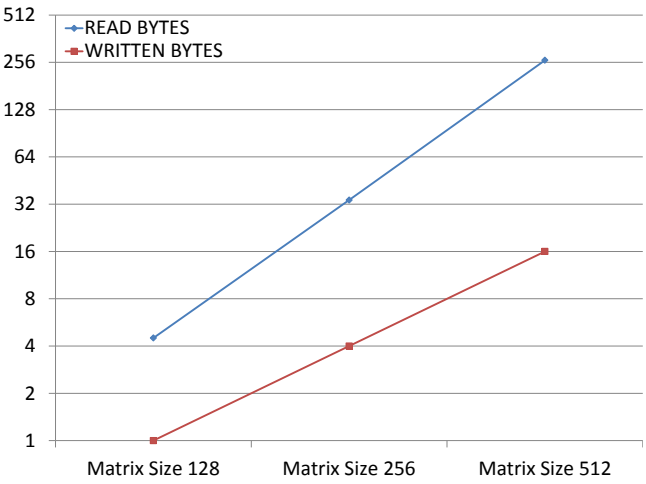


Figure 7: Scaling of Read and Write Data sets for DF-Threads. We present the case for Matrix Multiplication and we use `128x128` as baseline. This behaviour is same for the cases of 1, 2 and 4 nodes.

ZC706 FPGA boards, connected with SMA cables. Table 3 breaks down the reconfigurable resources utilization; LUTs and FFs utilization is less than 5% and 3% respectively. The Aurora IP configuration generated the user clock at 156.25 MHz that was also used to clock the DMA engine.

The PS7 run on bare-bone software a set of experiments for single and round-trip data transmissions between the two boards, ranging from 4 to 256 KBytes. Figure 6 shows the achieved throughput, which can be up to 490 MB/sec or 3920 Gbits/sec (single trip); a processing node with 4 MGTs can have a network throughput of up to 15.6 Gbits/sec.

4.4 DF-Threads Initial Results

In this sub-section, we report our experimental results on the read-write operation. For the experiment, we have used COTSon simulator [28].

In this case, the execution model is based on the DF-Threads (sub-section 3.4). For the evaluation, we have used the blocked matrix multiplication (Figure 2). The overhead to manage more DF-Threads is negligible even in small size

matrix multiplication. The parallelization is based on the ratio between the matrix size `n` and the block size `b`, i.e., the expected number of DF-Threads is `n/b` and the number of instructions increase as $O(n^3)$. We present the experiment in Figure 7, we consider three situations: `n=128, 256, 512` while the block size is fixed to `b=8`. In Figure 7, we can see the total amount of read and write data set size (bytes). The read and write data set size follows a linear trend and is also not creating a saturation effect as we use one, two and four nodes.

5 Conclusion

The AXIOM platform aims at integrating multiple heterogeneous SoC (currently with an FPGA) board by its new high-performance connection link and the task-based OmpSs programming model which can afford single and multiple-node heterogeneous parallel execution, transparently to the programmer. In this paper, we present the interconnects developed for the AXIOM system. The initial results show good potential for implementing a dataflow-based execution and easy programming model.

6 Acknowledgment

This work is partially supported by the European Union H2020 program through the AXIOM project (grant ICT-01-2014 GA 645496) and HiPEAC (GA 687698), by the Spanish Government through Programa Severo Ochoa (SEV-2015-0493), by the Spanish Ministry of Science and Technology through TIN2015-65316-P project and by the Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272).

References

[1] J. Lee, B. Bagheri, and H.-A. Kao (2015), *A cyber-physical systems architecture for industry 4.0-based manufacturing systems*, Manufacturing Letters, vol. 3, pp. 18–23.

[2] S. Mazumdar, E. Ayguade, N. Bettin, J. Bueno, S. Ermini, A. Filgueras, D. Jimenez-Gonzalez, A. Martinez, X. Martorell, F. Montefoschi, D. Oro, D. Pnevmatikatos, A. Rizzo, D. Theodoropoulos, and R. Giorgi (2016), *Axiom: A hardware-software platform for cyber physical systems*, DSD 2016, 19th Euromicro Conference on Digital Systems Design (DSD), pp. 539–546.

[3] C. Alvarez, E. Ayguade, J. Bosch, J. Bueno, A. Cherkashin, A. Filgueras, D. Jiminez-Gonzalez, X. Martorell, N. Navarro, M. Vidal, D. Theodoropoulos, D. N. Pnevmatikatos, D. Catani, D. Oro, C. Fernandez, C. Segura, J. Rodriguez, J. Hernando, C. Scordino, P. Gai, P. Passera, A. Pomella, N. Bettin, A. Rizzo, and R. Giorgi (2016), *The axiom software layers*, Microprocessors and Microsystems.

[4] C. Alvarez, E. Ayguade, J. Bueno, A. Filgueras, D. Jimenez-Gonzalez, X. Martorell, N. Navarro, D. Theodoropoulos, D. N. Pnevmatikatos, D. Catani, et al., *The axiom software layers*, Digital System Design, 2015 Euromicro Conference on, pp. 117–124, IEEE.

[5] D. Theodoropoulos, D. Pnevmatikatos, C. Alvarez, E. Ayguade, J. Bueno, A. Filgueras, D. Jimenez-Gonzalez, X. Martorell, N. Navarro, C. Segura, et al., *The axiom project (agile, extensible, fast i/o module)*, Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2015 International Conference on, pp. 262–269, IEEE.

[6] P. Burgio, C. Alvarez, E. Ayguad, A. Filgueras, D. Jimnez-Gonzlez, X. Martorell, N. Navarro, and R. Giorgi (2015), *Simulating next-generation cyber-physical computing platforms*, Ada User Journal, vol. 36, no. 4, pp. 259–263.

[7] Xilinx Inc. (2015), *Zynq series*.

[8] A. Duran, E. Ayguadé, R. M. Badia, J. Labarta, L. Martinell, X. Martorell, and J. Planas (2011), *Ompss: a proposal for programming heterogeneous multi-core architectures*, Parallel Processing Letters, vol. 21, no. 02, pp. 173–193.

[9] J. Bueno, X. Martorell, R. M. Badia, E. Ayguadé, and J. Labarta (2013), *Implementing ompss support for regions of data in architectures with multiple address spaces*, Proceedings of the 27th international ACM conference on International conference on supercomputing, pp. 359–368, ACM.

[10] J. Balart, A. Duran, M. Gonzàlez, X. Martorell, E. Ayguadé, and J. Labarta (2004), *Nanos mercurium: a research compiler for openmp*, Proceedings of the European Workshop on OpenMP, vol. 8, p. 56.

[11] D. Bonachea (2002), *Gasnet specification, v1. 1*.

[12] L. Rizzo (2012), *Netmap: a novel framework for fast packet i/o*, 21st USENIX Security Symposium (USENIX Security 12), pp. 101–112.

[13] Xilinx Inc. (2014), *8b/10b protocol specification*.

[14] R. Giorgi and A. Scionti (2015), *A scalable thread scheduling co-processor based on data-flow principles*, Future Generation Computer Systems, vol. 53, pp. 100–108.

[15] L. Verdoscia, R. Vaccaro, and R. Giorgi (2014), *A clockless computing system based on the static dataflow paradigm*, Proc. IEEE Int.l Workshop on Data-Flow Execution Models for Extreme Scale Computing, pp. 30–37.

[16] M. Solinas, M. Badia, F. Bodin, A. Cohen, P. Evripidou, P. Faraboschi, B. Fechner, G. Gao, A. Garbade, S. Girbal, D. Goodman, B. Khan, S. Koliaï, F. Li, M. Lujàn, A. Mendelson, L. Morin, N. Navarro, A. Pop, P. Trancoso, T. Ungerer, M. Valero, S. Weis, S. Zuckerman, and R. Giorgi (2013), *The teraflux project: Exploiting the dataflow paradigm in next generation teradevices*, 19th Euromicro Conference on Digital Systems Design (DSD), pp. 272–279, 2013.

[17] A. Portero, Z. Yu, and R. Giorgi (2011), *Teraflux: Exploiting tera-device computing challenges*, Procedia Computer Science, vol. 7, pp. 146–147.

[18] R. Giorgi (2015), *Scalable embedded systems: Towards the convergence of high-performance and embedded computing*, Embedded and Ubiquitous Computing (EUC), 2015 IEEE 13th International Conference on, pp. 148–153, IEEE.

[19] K. P. Gostelow, W. Plouffe, et al. (1977), *Indeterminacy, monitors, and dataflow*, ACM SIGOPS Operating Systems Review, vol. 11, pp. 159–169, ACM.

[20] K. M. Kavi, R. Giorgi, and J. Arul (2001), *Scheduled dataflow: Execution paradigm, architecture, and performance evaluation*, IEEE Transactions on Computers, vol. 50, no. 8, pp. 834–846.

[21] R. Giorgi (2016), *Exploring dataflow-based thread level parallelism in cyber-physical systems*, Proceedings of the ACM International Conference on Computing Frontiers, pp. 295–300, ACM.

[22] S. Weis, A. Garbade, B. Fechner, A. Mendelson, R. Giorgi, and T. Ungerer (2016), *Architectural support for fault tolerance in a teradevice dataflow system*, International Journal of Parallel Programming, vol. 44, no. 2, pp. 208–232.

[23] R. Giorgi and P. Faraboschi (2014), *An introduction to df-threads and their execution model*, Computer Architecture and High Performance Computing Workshop, International Symposium on, pp. 60–65, IEEE.

[24] S. Weis, A. Garbade, J. Wolf, B. Fechner, A. Mendelson, R. Giorgi, and T. Ungerer (2011), *A fault detection and recovery architecture for a teradevice dataflow system*, Proc. IEEE Int.l Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM), pp. 38–44.

[25] L. Verdoscia and R. Giorgi (2016), *A data-flow soft-core processor for accelerating scientific calculation on FPGAs*, Mathematical Problems in Engineering, vol. 2016, pp. 1–21.

[26] B. S. Center (2016), *Extrae instrumentation library*. (Accessed June 16, 2016).

[27] V. Pillet, J. Labarta, T. Cortes, and S. Girona (1995), *Paraver: A tool to visualize and analyze parallel code*, Proceedings of WoTUG-18: Transputer and occam Developments, vol. 44, pp. 17–31, mar.

[28] E. Argollo, A. Falcón, P. Faraboschi, M. Monchiero, and D. Ortega (2009), *Cotson: infrastructure for full system simulation*, ACM SIGOPS Operating Systems Review, vol. 43, no. 1, pp. 52–61.

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