ADA USER JOURNAL

Volume 23
Number 2
June 2002

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 first 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 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 any of the 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.
- News and miscellany of interest to the Ada community.
- Reprints of articles published elsewhere that deserve a wider audience.
- Commentaries on matters relating to Ada and software engineering.
- Announcements and reports of conferences and workshops.
- Reviews of publications in the field of software engineering.
- Announcements regarding standards concerning Ada.

Further details on our approach to these are given below.

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

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. Since not all of our readers have access to resources such as the World Wide Web and Usenet, or have enough time to search through the information that can be found in those resources, we 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 to the editor. Electronic submission is preferred – typed manuscripts will only be accepted by the Editor by prior arrangement.

Prospective authors are encouraged to contact the Editor by email to determine the best format for submission. Contact details can be found near the front of each edition.

Example papers conforming to formatting requirements as well as some word processor templates are available from the editor. There is no limitation on the length of papers, though a paper longer than 10,000 words would be regarded as exceptional.

Editorial

Once again there is much interesting Ada related news to report. The first news article refers to the Ada Rapporteur Group (ARG) and its call for APIs for the 2005 revision of Ada. This is also covered in a separate announcement later in the issue.

Thanks are due to John Barnes, who was able to edit the news for this issue in the absence of regular news editor, Dirk Craeynest.

Finally, this issue marks the end of my tenure as Editor-in-Chief. I am pleased to announce that Tullio Vardanega will be taking over the position for the next issue and wish him well.

Neil Audsley
York
June 2002
Email: Neil.Audsley@cs.york.ac.uk

News

Dirk Craeynest (ed)

Offis nv/sa and K U Leuven. Email Dirk.Craeynest@offis.be

Contents

Ada-related Organizations

ARG calls for proposals

From owner-team-ada@ACM.ORG
Date: Fri May 24 14:10:31 2002
Subject: ARG asks Ada Community for API Proposals.
To: TEAM-ADA@ACM.ORG

The Ada Rapporteur Group (ARG) is the technical committee in charge of proposing amendments to the language to WG9, the ISO working group on Ada. The ARG has begun work on the next revision of Ada, planned for 2005. As part of this revision, there has been a lot of interest in the Ada community for the standardization of reusable components and APIs to existing services.

While the ARG will conduct (based on input from the Ada community) the revision of the core language and annexes, it doesn't have the resources to develop proposals itself for the standardization of reusable components or APIs. Standardization of components or APIs often will best be accomplished with secondary standards rather than part of the core language standard. The ARG will oversee the development of such secondary standards, but this is best accomplished by cooperating with external groups developing the substance of such standards.

Therefore, the ARG would like to ask the Ada community to submit proposals for the standardization of APIs. Proposals must include (at least) a set of Ada specifications, and a semi-formal description of the semantics of each declaration, such as can be found in the annexes of the Reference Manual. Developing such proposals usually will require the formation of formal or informal working groups.

The ARG will evaluate these proposals using a variety of criteria. For a more detailed version of this announcement, including a (partial) list of evaluation criteria, please see: http://www.adaic.org/news/call4apis.html

Randall Brukardt Editor, ISO/IEC JTC1 SC22 WG9 ARG

[and here are the details -- jb]

ISO Working Group asks Ada Community for Candidate APIs for Standardization

From Pascal Leroy, Principal Software Engineer, Rational Software Corp., Chair, Ada Rapporteur Group

[Please also see page 96 in this edition of the AUJ – Editor]

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

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

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

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

* Benefits of the standardization. Presumably the advantage of standardization is that it brings uniformity and portability among implementations. However, there is a significant overhead associated with a formal standardization process, so in some cases a de facto standard may bring practically the same benefits at a much lower cost.

* Usefulness of the API. APIs which have been conjured up solely for the purpose of writing a proposal, or which have been used by a very small group of users, are less likely to be generally useful than APIs which have been available for years and have benefited from feedback from a large user base.

* Quality and precision of the proposal. At a minimum, the proposal must include a set of Ada specifications, and a semi-formal description of the semantics of each declaration, such as can be found in the annexes of the Reference Manual. A rationale showing examples of use, explaining the choices that were made, the alternatives that were considered, and why they were discarded, would also be much appreciated.

* Community consensus for the proposal. Proposals with a substantial consensus of the Ada community or the appropriate subcommunity are preferred over proposals made by an individual or small group. This is not to say that a proposal primarily authored by an individual is necessarily bad (indeed, it is likely to provide a more consistent proposal), but to encourage authors to seek input/approval from as many potential users of the API as possible.

* Portability and language usage. The definition of the API must not depend on implementation-defined characteristics of a particular compiler, although it is acceptable to require the compiler to support some Specialized

Needs Annex (or part thereof). As much as possible, the API should only use the features of Ada 95 (as opposed to those that are under consideration for the 200Y amendment) although we realize that this may not be practical in some cases.

* Implementation. A publicly available reference implementation would be useful, although this is not a strict requirement, as in some cases that may cause intellectual property issues.

* Test suite. A test suite ensuring conformity to the specification should be provided at some point during the standardization process. This is especially important for standards for which no publicly available reference implementation will be available. This doesn't necessarily mean that there will be a formal conformity assessment process like there is for compilers, but it will help implementers ensure that they comply with the standard.

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

Ada Semantic Interface Specification (ASIS)

Ada Browse

From: Clyde Roby <roby@ida.org>
Date: Tue, 12 Mar 2002 07:13:32 -0500
Subject: New ASIS product: AdaBrowse - a javadoc for Ada 95
To: SIGADA-ASIS@ACM.ORG

As you all know, on our web page at:
http://www.acm.org/sigada/wg/asiswg/asiswg.html
we list several ASIS-based products, with detailed descriptions on
http://www.acm.org/sigada/wg/asiswg/ASIS_Clients.html

Dr. Thomas Wolf (twolf@acm.org and http://home.tiscalinet.ch/t_wolf/) has developed his own, free (GPLed) utility called AdaBrowse; I've added it to these lists. Here is a succinct description of AdaBrowse. I invite you all to visit our web page to see the full description and to download the program.

AdaBrowse is an HTML generator for Ada 95 library unit specs. It generates structured and fully cross-referenced HTML documentation from Ada 95 sources, similar to what javadoc does for Java.

AdaBrowse is an ASIS application: it uses ASIS to produce precise cross-references in the generated HTML, and to extract semantic information to structure the generated HTML and to generate e.g. a type index containing also the primitive operations of the types.

It works with both GNAT 3.13p and 3.14p and possibly with other ASIS implementations, too. It runs on Windows NT/2000, and also (hopefully; I don't have a Unix-box!) on Unix-like systems.

AdaBrowse is available at the URL:
http://home.tiscalinet.ch/t_wolf/tw/ada95/adabrowse/
in source and executable (for GNAT 3.14p on Win NT/2000) form. A brief description of AdaBrowse is in the appendix. Visit the ASISWG website and see more about AdaBrowse!

Clyde Roby, ASISWG Webmaster

From: Thomas Wolf <t_wolf@angelfire.com>
Date: Tue, 7 May 2002 11:36:34 +0200
Subject: ANN: AdaBrowse 2.0
Newsgroups: comp.lang.ada

AdaBrowse is a HTML documentation generator for Ada 95 library unit specifications. It is distributed under the GPL and is available at the URL:
http://home.tiscalinet.ch/t_wolf/tw/ada95/adabrowse/
It is available as both a pre-built executable (for Win NT/2k and GNAT 3.14p), and the sources.

AdaBrowse builds and runs without further ado on both Win NT/2k and Unix systems, and shouldn't be too hard to port to other ASIS implementations than ASIS-for-GNAT.

For those who have never heard of AdaBrowse yet: it is like a javadoc for Ada 95, but much more versatile and powerful than javadoc for Java.

The distribution comes with a comprehensive user's guide including examples and instructions on how and where to submit bug reports.

AdaBrowse differs from gnathtml in several ways:

- AdaBrowse is a stand-alone executable; gnathtml is a perl script.
- AdaBrowse is more flexible, and can be customized to a much greater extent.
- AdaBrowse produces structured HTML output including formatted descriptions; gnathtml basically just encloses the source in <PRE></PRE> tags and adds cross-references.
- AdaBrowse uses ASIS to collect cross-reference information, whereas gnathtml relies on the GNAT-specific cross-reference info in the ALI files.
- AdaBrowse uses ASIS to gather semantic information about library units and makes use of it, e.g. to find all the primitive operations of a type.
- AdaBrowse can call e.g. a compiler if no ASIS information is found, whereas gnathtml doesn't generate cross-references if no ALI file is found.
- AdaBrowse does some limited form of pretty-printing, such as using identifiers as cased in their definition everywhere.

V2.0 is a major upgrade over the last announced version 1.5:

- Some more work-arounds for bugs in ASIS-for-GNAT 3.14p.
- It now supports user-defineable HTML elements. AdaBrowse macro-replaces such user-defined elements; recursive definitions give an error. User-defined HTML elements are specified through keys in configuration files.
- Configuration files can include other configuration files. Recursive inclusion is detected and gives an error message.
- On some keys in configuration files, AdaBrowse now does environment variable substitution (using the bash syntax).

Thomas Wolf

Ada-related Resources

Updated Ada Conformity Asesment Test Suite

From: "Technical Webmaster" <Webmaster@adaic.com>
To: <Announce@adaic.com>
Date: Tue, 2 Apr 2002 19:17:41 -0600
Subject: [AdaIC] ACATS 2.5 Released

The ACAA has released an update to the Ada Conformity Assessment Test Suite (ACATS). The new version, version 2.5, includes new tests and modifications from the last year. The new tests increase the coverage of the test suite to include a larger portion of the changes in Technical Corrigendum 1, the recently published corrections to the Ada standard.

ACATS 2.5 is available from the AdaIC web site at
http://www.adaic.org/compilers/testing.html

Ada-related Tools

Simple Graphics

From: Michael Gonzalez <mgh@unican.es>
Date: Wed, 20 Mar 2002 11:29:55 +0100
Organization: Universidad de Cantabria
To: "Dirk Craeynest (Ada Belgium)" <Dirk.Craeynest@cs.kuleuven.ac.be>
Subject: Win_IO

Dirk: You may find this product interesting for the Ada news in AUJ. Michael.

Win_IO is a set of packages for graphical input and output. It is designed specially for students or Ada users who do not want to spend their time learning a complex graphical user interface, but who are "tired" of the old-fashioned text-oriented input and output. Win_IO has the same goals as JEWL (John English Windows Library, http://www.it.bton.ac.uk/staff/je/jewl/), but is simpler (and less powerful) and is portable within Unix, Linux and Windows platforms. JEWL is currently only provided for Windows.

Win_IO is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation. It is based on GtkAda and Gnat.

Win_IO is composed of the following modules:

* Input_Windows: Provides a simple window with I/O capabilities for data of the types Integer, Float, and String. Several data can be displayed and/or retrieved on the same window.
* Output_Windows: Provides a simple window with Output capabilities for data of the types Integer, Float, and String. Several data can be displayed on the same window.
* Message_Windows: Provides a simple window for displaying a short message. It provides an OK button for closing the window.
* Menu_Windows: Provides a simple window with several buttons that enable the user to select from a number of options. It is a generic package that must be instantiated with an enumeration type. One button will be created for each value in this type.
* Graphics_Windows: Provides a simple window with drawing capabilities.
* Plot_Windows Provides a simple window for drawing two-dimensional graphs from sets of points.

You can find Win_IO in: http://ctrpc17.ctr.unican.es/win_io/

Michael Gonzalez Harbour, Dpto. de Electronica y Computadores, Universidad de Cantabria, Avda. de los Castros s/n, E-39005 Santander, SPAIN, Phone: +34-942-201483, Fax: +34-942-201402

Ada Web Server

From: Pascal Obry <p.obry@wanadoo.fr>
Date: 29 Apr 2002 18:45:43 +0200
Subject: ANNOUNCE: AWS 1.2
Newsgroups: comp.lang.ada

Dmitriy Anisimkov and I are very happy to announce the availability of the AWS 1.2 release. The API could change slightly at this stage but should be fairly stable now.

AWS stand for Ada Web Server. It is not a real Web Server like Apache. It is a small yet powerful HTTP component to embedded in any applications. It means that you can communicate with your application using a standard Web browser and this without the need for a Web Server. AWS is fully developed in Ada with GNAT.

AWS support SOAP, Server Push, HTTPS/SSL, client HTTP, hotplug modules. We have worked very hard to make this release as stable as possible. Note that Hotplug modules are very nice but have a potentially security hole as it is implemented today. A new secure implementation will be proposed in a future version.

The SOAP implementation has been validated on: http://validator.soapware.org/

Here are the main changes:

- You need GNAT 3.14 to build AWS 1.2 (GNAT 3.13 is not supported anymore).
- Add a main procedure termination controller (AWS.Server.Wait)
- Fix some memory leak in AWS. Response.Data and AWS.Server. Protocol_Handler for binary data.
- In AWS.URL, function URI was not correctly named. It has been renamed Pathname. This is a backward compatibility problem. Path and File function has been added into AWS.URL.
- Fix bug to close a connection when server is heavy loaded.
- Add AWS.Services.Page_Server service. This service is a straight forward implementation of a simple static web page server. See WPS demo. It supports two template files: 404.thtml and aws_directory.thtml.
- Fix race condition in AWS.Server implementation. This was a very nasty bug, sockets could be handled in two different slots. If you are experiencing bug with heavy loaded servers you should plan to upgrade as soon as possible.
- Add dispatchers facilities which is more general than the callback procedure (access to procedure) for example it can transport user's data. This is the base of a general framework for high level services.
- Add three high level Dispatcher facilities (AWS.Services.Dispatchers): 1) on URI, 2) on request method, 3) on Host name (also called virtual hosting)
- Add AWS.Templates (renaming of Templates_Parser) as this component is a very important one for Web development.
- AWS can now have servers binded to different IP addresses if the computer has more than on IP addresses. See AWS.Config.Server_Host.
- New version of libssl32.dll and libeay32.dll based on OpenSSL 0.9.6c.
- Client handle properly the HTTP continue response message.
- Templates_Parser now integrated into AWS.Templates package. This version has a cache fully is thread safe.
- Session cookie was set for first path (and sub path) used, it means that it was possible to have multiple session for a Web site. This behavior was the result of a bug. Now a single session is created for the whole site (starting at /).
- Fix timeouts for client keep-alive connection.
- SOAP handle properly zero length array.
- SOAP handle properly Array of Record.
- Boolean types are now directly handled on sessions.
- Now always install AWS under directory AWS, INSTALL make variable must point to the AWS parent directory.
- Plus many small fixes, enhancements and documentation work.

You can have a look at docs/TODO file to see what are the topics that we will probably implement in future releases.

NOTE: Since we have switched to the .PNG file format we have found that Netscape Navigator is not able to display the PNG transparent layer properly!

At this stage we feel that AWS is ready to build small to medium Web servers. AWS has been reported to work on Windows NT/XP, Linux and FreeBSD 4.1.

With this new version you'll need at least version 1.0 of the Socket binding >from ENST. See pointers below.

The OpenSSL libraries (optional) distributed are for Windows only. On UNIX you'll have to build the libraries from sources, it is quite easy to do so. This has been tested under Linux without trouble.

See documentation for build information.

AWS User's Mailing List: http://lists.act-europe.fr/mailman/listinfo/aws

AWS Home Page (sources and documentation): http://libre.act-europe.fr/

Templates_Parser sources: Templates_ Parser module (sources and documentation) is provided with AWS distribution. Latest version of this module and the documentation can be found at:

http://perso.wanadoo.fr/pascal.obry/contrib.html and http://perso.wanadoo.fr/pascal.obry/templates_parser.html

Templates_Parser is a useful add-on for AWS. You should have a look at it if you plan to develop a Web service. It Templates_Parser permits the separation of the HTML design from the Ada code.

Some other Templates engine are WebMacro, FreeMarker, PHP, ASP, JSP and Velocity. All of them are based on explicit iterators (#foreach with a variable) where Templates_Parser is based on implicit ones (you use a more intuitive table iterator). Be sure to check the documentation. Only the Velocity project has the goal to support complete separation of HTML design and code.

GNU/Ada - GNAT You need at least version 3.14 to use AWS 1.2: ftp://cs.nyu.edu/pub/gnat/

XMLada (optional): You need this library only if you want to use AWS SOAP feature. You need at least XMLada 0.7.1: http://libre.act-europe.fr/

Socket binding: Since AWS 1.2 you need at least version 1.0 of the Socket binding.
for Win32:
http://perso.wanadoo.fr/pascal.obry/contrib.html
http://vagul.tripod.com/adasockets.tgz
for UNIX:
http://www.rfc1149.net/devel/adasockets

POSIX Binding (optional). For Win32:
http://perso.wanadoo.fr/pascal.obry/contrib.html
for UNIX:
http://www.cs.fsu.edu/~baker/florist.html

OpenSSL library (optional). Sources for UNIX or Win32:
http://www.openssl.org
Binaries for Win32: included with the main AWS distribution (win32 directory).

Note that we have used and we distribute (for Win32 platform) OpenSSL version 0.9.6c with this AWS release. OpenSSL have been built with GCC version 2.95.3 with -O3 optimization level.

See OpenSSL license (docs/openssl.license).

Windows Services API (optional):
To build the runme demo as a Windows NT/2000 services you must download the services API made by Ted Dennison for his SETI@Home project:
http://www.telepath.com/dennison/Ted/SETI/SETI_Service.html

You can report bugs to:
Dmitriy Anisimkov (anisimkov@yahoo.com) or Pascal Obry(p.obry@wanadoo.fr)

It would be nice if you could also sent us a note if you are using AWS just to know if it is used at all or not :) And if you are ok, we'll add an entry for your project in the next section.

AWS User's Mailing List:

A good way to keep informed of AWS news and to share experiences with other AWS users is to register to the AWS dedicated mailing list. See:
http://lists.act-europe.fr/mailman/listinfo/aws

AWS uses

- SETI@Home from Ted Dennison.

AWS is used as a "plugable" GUI to retrieve different program status.
- DOCWEBSERVER from Wiljan Derks
In our department we keep our documents in a directory tree. These documents are all project related and have a certain naming convention to be able to find the right document. In the past I already wrote a program that searches though this directory and then converts the found documents into fixed html pages. With AWS I was able to get a much nicer setup. I have now a server that can do the following: - browse through the projects in explorer style. The html contains info about the document like date and title. - one can check in documents through the web interface - it shows our download page as I have send you in the example - we have now all our documentation in small pieces of html as is needed to build .chm (w2k compiled help) files. For these we use a content file, that is also stored in the document archive.
The docwebserver gives by reading all this stuff the direct view on this documentation. On the other hand I can run some tool and automatically generate the .chm files.
- OESM Server (OESM=Overall Equipment Status Monitoring) from Wiljan Derks
I am working on a project now for our factories. ITEC mainly delivers equipment for discrete semiconductor assembly. Almost all of that equipment is now controlled by a similar Ada 95 based code with having a lot of code in common. One of the common things, is the way we log errors and state changes of our equipment.

The OESM Server is an application which copies all this information continuously to its local pc by opening the proper files on the remote equipment. That data copied is also stored in local files. The web server component of the application can then, making use of that data, give reports that show things like the amount of products produced in a certain period, error paretos of equipment, mtbf, %time in production and of course many other things.

The cool thing of course is that this information can easily be charted (I am use kavachart) and it allows simple navigation through different groups of equipments and different views on the equipment.

- WORM from Pascal Obry see:
http://www.ada-france.org/ADHERENTS/101100/05-obry.pdf
A Web server to share bookmarks, this server was using a standard CGI design. To keep session information we were using a GLADE partition. With AWS the design has been really simplified, there is no need for a session partition, there is no need to build all CGI as partitions too. GLADE is now used only to handle distributed objects. Indeed WORM is a multi-server system (using RACW) with a register/unregister mechanism.

Also the server seems to be fastest, there is no more CGI to spawn.

- Internet Currency Trading System by: Dmitriy Anisimkov at http://www.actforex.com

This is a server is used to keep historical data about currency trading to build charts of currency prices. The charts viewer part is written in Java and loaded through AWS. This server can be reach on the Internet.

Ongoing work is done to based this development on AWS framework only and to remove all the Java layers. It is also interesting to note that this is an heavy loaded server, it handle something like 40 to 50 requests per seconds on a Windows 2000 Server.

- http://www.forexcoach.com site is powered by AWS. This site has been done by Dmitriy Anisimkov.
Thanks to all who have reported bugs and have sent us patches.
Dmitriy & Pascal.

--| Pascal Obry (Team-Ada Member) 45, rue Gabriel Peri - 78114 Magny Les Hameaux FRANCE

From: anisimkov@yahoo.com (Dmitriy Anisimkov)
Date: 20 Apr 2002 19:08:20 -0700
Subject: Ada95 e-Business
Newsgroups: comp.lang.ada

I want to show the Ada95 usage in the public e-Business solution.
On the Site:
http://www.actforex.com/frames.html
In the left menu choose "Demo", then in the top of the page choose "Registration"
The registration form will be handled by the HTTP Server written in Ada95 (information about the Ada Web Server is http://libre.act-europe.fr/aws/).

You are going to receive the email with login_id and password for the access to the forex demo system.

If you don't want to register, i.e. betray your email address. you can use

Demo Username: ttest Demo Password: 348

On the top of the page with registration success message choose "Demo Login". Choose the language for login. Enter the name and password received by email.

The client application is using the Sun JAVA Plug-in 1.3, if it is not already installed, it is going to install during first applet loading. In the applet menu you can choose the Help/About You can see the http server version over there. It is Ada Web Server. You can choose the File/Reports from applet menu, and see reports dynamically generated in the RDBMS and published by AWS.

You can choose the View/Charts from applet menu, and see the Forex charts. Data for the charts generated in the RDBMS, published by AWS, and displayed by Java applet. Other usage of the AWS is the complete site http://www.forexcoach.com
If you don't want to register, you can use the login id : Dima password: aws

Motif Binding

From: vgodunko@vipmail.ru (Vadim Godunko)
Date: 7 Mar 2002 13:45:44 -0800
Subject: Announce: AdaBindX 0.7.2 released
Newsgroups: comp.lang.ada

New version of Ada/X/Motif bindings named adabindx you may find at: http://home.arcor.de/hfvogt/programming.html

Win32POSIX

From: Pascal Obry <p.obry@wanadoo.fr>
Date: 09 Mar 2002 15:15:53 +0100
Subject: Win32POSIX v1.12b
Newsgroups: comp.lang.ada, fr.comp.lang.ada

I have just uploaded a new version of Win32POSIX. A POSIX implementation for Windows based system. This is just a partial implementation since Windows is really far to be POSIX compliant :)

Thanks a lot to Jean-Pierre.Rosen ;) for his contributions. Jean-Pierre as continued the foolish project to have a POSIX interface on native (read non Cygwin based) Windows :) and we can tell you that it is quite difficult! Win32 API is just plain amazing sometimes :)

Anyway you'll find v1.12b on my homepage. A direct access to the Win32POSIX page is: http://perso.wanadoo.fr/pascal.obry/archive/w32posix.html

Here are the main changes since 1.11b

* POSIX API should be thread safe. In many places this was not true before.
* POSIX.Process_Primitives It is possible to launch .com and .exe (not only .exe as before).
* POSIX.Calendar Handle milliseconds.
* POSIX.Files Is_Symbolic_Link added. Is_Socket added. Both always return False on Win32. For_Every_Directory Entry, check for pathname with directory separator.
* POSIX.File_Status Get_File_Status new version should be better than before. Do not handle devices as this seems to be impossible on Win32.
* POSIX.Process_Environment Environment_Value_Of correctly return Undefined if variable not found.

Pascal.

AdaSockets binding

From: Pascal Obry <p.obry@wanadoo.fr>
Date: 09 Mar 2002 15:20:57 +0100
Subject: [ANNOUNCE] Adasocket v1.0 port to Win32
Newsgroups: comp.lang.ada,fr.comp.lang.ada

Dmitriy Anisimkov has ported to Win32 the latest version of the AdaSockets binding (v1.0). You can download it from:
http://vagul.tripod.com/adasockets.tgz
http://perso.wanadoo.fr/~pascal.obry/contrib.html

Next AWS version (v1.3) will need this AdaSockets version.

Pascal.

Interfacing to Prolog

From: Dale Stanbrough <dstanbro@bigpond.net.au>
Date: Fri, 12 Apr 2002 14:03:10 GMT
Subject: Re: How to interface to Prolog?
Newsgroups: comp.lang.ada

> Adrian Hoe wrote: I would like to call a procedure written in Prolog from Ada or another way round. I Could not find any information on this. One way to resolve this is to write a C-wrapper but I want to do it pure Ada-Prolog if possible. Any ideas, anyone?

There is a Prolog interpreter written in Ada from the Anna (Ada annotation language) toolset written at Stanford (?) many years ago, that I made more Ada95 like quite some time ago.

It was for a project that never really got off the ground, so although the packages seem to work a bit, I can't guarentee how thoroughly it would work.

I attempted to contact the copyright holders when I got a copy of it, but could never make raise them from the dead.

The source code can be found at http://goanna.cs.rmit.edu.au/~dale/software/index.html

Dale

From: Anatoly Chernyshev <rhezusfactor@yahoo.com>
Subject: Re: How to interface to Prolog?
Newsgroups: comp.lang.ada

There is very good yet freely available Prolog package at www.amzi.com, which includes dll-libraries and several samples of how to interface Amzi Prolog with other languages (C, Java, VB...) You may wish to peep into these samples and adapt the library for Ada. I was planning this for myself a while ago, but had no time yet.

Regards, Anatoly

Database system in Ada

From: mkudvin@atlas.cz (Matthew Goodwin)
Date: 22 Feb 2002 13:32:55 -0800
Subject: GPL Enterprise Database System in Ada
Newsgroups: comp.lang.ada

I have just released a GPL Enterprise (Accounting, Order Entry, Shipping, Inventory) database system (aka Zephyr Basecamp). It is written in Ada, uses the Postgresql database and Gtk. It, thanks to Ada, works on both Windows and Linux.

Relevant comments would be greatly appreciated.

Matthew Goodwin

Ada-related Products

ACT announces GPS

URL: http://www.gnat.com/texts/news/news_gps.htm

Ada Core Technologies announces GPS: a new developer-friendly programming Environment

A customizable, multi-platform system with advanced navigation support

New York, N.Y., April 30, 2002, 9:00 a.m. - Ada Core Technologies, Inc., the leader in GNAT and Ada 95 technology, today announced Q4 availability of GPS, the GNAT Programming System. GPS is a developer-friendly Ada, C, and C++ graphical programming system for native and embedded software development. GPS is easy to learn and use, yet it allows programmers to develop, build, and maintain large complex systems.

"The GPS IDE reflects the move towards less centralized software project organization and allows for maximum flexibility without compromising reliability or commonality of style", said Robert Dewar, President of Ada Core. "GPS was designed by programmers for programmers", added GPS project leader Arnaud Charlet. GPS offers a compelling graphical interface that integrates a syntax-oriented editor, a source-level debugger, source code navigation, dependency and unit hierarchy browsers, call graphs, a project configuration manager, project dependency graphs, unit testing, and version control support, among other tools. GPS can be tailored by users to integrate their preferred tools, such as editors, use their favorite configuration management, add additional tools such as change tracking, or add support for additional programming languages.

Combined with the GNAT Pro Ada 95 and C tool suites, GPS offers a sophisticated environment for native development on GNU/Linux, UNIX, and Windows, or embedded development for VxWorks and LynxOS.

For additional product information please visit the Ada Core website at
http://www.gnat.com
or contact sales@gnat.com.
or contact Nancy Cruz Ada Core Technologies, Inc. (212) 620-7300 Ext. 117 cruz@gnat.com

ACT and Compaq

URL: http://www.gnat.com/texts/news/news_compaq.htm

Ada Core Technologies awarded Compaq contract for porting Ada Toolset to Compaq OpenVMS for Itanium Processor Family

GNAT Pro Ada 95 Toolset and Compaq's OpenVMS Provide Solid Foundation for Software Development

New York, N.Y., April 30, 2002, 9:00 a.m. - Ada Core Technologies, the leader in GNAT and Ada 95 technology is pleased to announce that it has been awarded a contract by Compaq Computer Corporation to implement the GNAT Pro Ada 95 Tool Suite on the Compaq OpenVMS operating system for the Itanium Processor Family.

This contract involves the provision of an Ada 95 compiler, an accompanying toolset integrated into the OpenVMS development platform, and project support.

"Ada on OpenVMS is an excellent match", observed Ada Core's President and CEO Robert Dewar, "Over the years, both Ada and Compaq's OpenVMS operating system have enjoyed well-deserved reputations for reliability, and we expect that the port of GNAT to the new Compaq OpenVMS on Itanium platform will give users a solid foundation for software development."

The port of GNAT Pro Ada 95 is expected to play a key role in the port of Compaq OpenVMS to the Itanium Processor Family. This port is expected to be upwards compatible with both Compaq Ada (Ada 83), and the existing port of GNAT Pro for OpenVMS on Compaq AlphaServer systems. It will provide 100% full Ada 95 functionality including relevant special needs annex support..

"We are pleased to have such a respected industry leader as Ada Core Technologies working with us, "states Mark Gorham, vice president of Compaq's OpenVMS Systems Group. "With their expertise, we expect to be well positioned to continue our commitment to our key government and private sector customers."

For additional product information please visit the Ada Core website at
http://www.gnat.com
or contact sales@gnat.com.
or contact Nancy Cruz Ada Core Technologies, Inc. (212) 620-7300 Ext. 117 cruz@gnat.com

ACT announces GNAT 3.15a

From: "Cyrille Comar" <comar@ACT-Europe.FR>
Date: Mon, 25 Feb 2002 13:12:17 +0100
Subject: [9610-004] Release 3.15 is available on 7 native platforms

We are happy to announce the immediate availability of the candidate GNAT 3.15a release for the following platforms: alpha-tru64 pa-hpux ppc-aix mips-irix sparc-solaris x86-linux x86-solaris (other major platforms will follow shortly) The distributions can be found in ftp://ftp.gnat.com/gnatpro/3.15

The release represents a large step forward in GNAT technology. The release incorporates over 180 documented problem corrections and 126 documented enhancements, optimizations, new pragmas, tools and compiler warnings. The project facility has improved substantially, thanks to the very successful beta program carried on in version 3.14. This facility is now integrated into the 3.15 release, and documented in the GNAT User's guide.

Debugging capabilities have improved significantly thanks to new versions of gdb & gvd, as well as compilation features such as the new -gnatVf switch, which in combination with the pragma initialize_Scalar improves substantially the detection of uninitialized variables. We are looking forward to hearing from you about this new release.

With Best Regards,
ACT Europe and Ada Core Technologies

I-Logix Launches Rhapsody in Ada

URL: http://www.ilogix.com/news/press_detail.cfm?pressrelease=2002_02_19_055040_36053pr.cfm

I-Logix Launches Rhapsody in Ada

UML-Compliant Visual Development Platform For Ada Speeds Development of Embedded Military/Aerospace Applications

February 19, 2002 - Embedded Systems, Nürnberg, Germany - I-Logix Inc., the premier provider of enterprise solutions for embedded applications development, has launched Rhapsody(r) in Ada^(r), the latest member of I-Logix' award-winning Rhapsody product family and the first visual application development platform for Ada programmers based on the standard Unified Modeling Language® (UML).

Designed and optimized for the development of embedded applications in the military, aerospace and transportation industries where the Ada language is used extensively, Rhapsody in Ada generates fully traceable and customizable production quality code that is readily deployable. I-Logix is working with leading military and aerospace suppliers to streamline the integration of Rhapsody in Ada into the development process for mission critical real-time embedded applications. Rhapsody in Ada enables programmers to work quickly, easily and efficiently in a visual environment that has been fine-tuned to their needs. By shifting the focus of work from coding and debugging to design, increasing the opportunity for design re-use, and simplifying communication among design team members, Rhapsody in Ada can save over 30% in development cycle time and dramatically improve time-to-market for delivering real-time embedded applications targeting the Ada language.

Rhapsody in Ada employs a standard UML modeling environment that is well understood by the real-time embedded software engineering community, but unlike developers using other languages, Ada programmers need a high level of code customization flexibility in order to deal with conformance requirements such as safety, verification and certification issues. Rhapsody in Ada has been architected to enable end-users to customize the style of the generated Ada code. This enables them to conform with the often strict and well-defined development standards associated with military and aerospace applications.

Through full integration with DOORS ®, Rhapsody in Ada also addresses the important issue of requirements traceability that the size and complexity of Ada software development projects invariably present. DOORS provides the ability to describe system functionality in a textual format and integration with Rhapsody in Ada ensures that, at any point in the design cycle, the requirements can be traced directly into the UML design models and ultimately into the source code.

Rhapsody is a visual application development platform designed to meet the challenges of real-time embedded software development. It allows real-time embedded software engineers to analyze, design, implement, and test UML-based applications graphically. Production-quality code is automatically generated as the design evolves and graphical animation allows design diagrams on the development host to be debugged before testing the software on the target. This capability enables reuse at the design level and compresses the overall development cycle. Rhapsody has an open architecture that enables interfacing to leading requirements traceability, configuration management, and testing tools. Rhapsody in Ada can also import system models from the I-Logix Statemate MAGNUMTM system design solution. In addition, Rhapsody includes an adaptable real-time framework that enables target code deployment to virtually any Real-Time Operating System (RTOS).

Commenting on the launch of Rhapsody in Ada, Neeraj Chandra, Senior VP of Marketing & Corporate Development said: "Our decision to develop an Ada version of Rhapsody has been driven entirely by customers who want an easier, friendlier and quicker way to generate Ada code and manage their Ada projects. In short Ada developers have been crying out for Rhapsody. By delivering Rhapsody in Ada we are the first in the industry to provide them with a production quality code generation strategy from standard UML."

About I-Logix

Founded in 1987, I-Logix is a pre-IPO software company that provides enterprise solutions for real-time embedded applications development in the growing pervasive computing market. I-Logix' solutions significantly compress systems and software development cycles while improving product quality. These products allow engineers to graphically model the behavior and functionality of their embedded systems, analyze and validate the system and automatically generate production quality code in a variety of languages. I-Logix uniquely integrates and associates the entire design flow from concept to code across an enterprise using both conventional and collaborative Web-enabled technology.

I-Logix is a member of the Object Management Group® (OMG), the Bluetooth SIG, the International Council of Systems Engineers (INCOSE), a founding member of the Embedded Linux Consortium and a co-author of the Unified Modeling Language® (UML). I-Logix is backed by Phillips Ventures BV, Needham Capital Partners, ABS Ventures, Commonwealth Capital Ventures, Gilde Investments, One Liberty Ventures and North Bridge Venture Partners. The company is headquartered in Andover, Mass., and has sales offices and distributors throughout the USA, Europe and the Far East. I-Logix can be found on the Internet at http://www.ilogix.com.

[…]

Contacts: Carrie Kirby Public Relations Coordinator I-Logix 978-682-2100 carrie@ilogix.com Keith Mason Harvard Public Relations +44 (0) 20 8759 0005 keith@harvard.co.uk

* Editors notes: Rhapsody and Statemate are registered trademarks of I-Logix Inc. Rhapsody in MicroC, Statemate MAGNUM and ROPES are trademarks of I-Logix Inc. OMG marks and logos are trademarks or registered trademarks, service marks and/or certification marks of Object Management Group, Inc. registered in the United States.

(c)2001 I-Logix Inc. All rights reserved. Three Riverside Drive Andover, Ma.01810 Tel: +1-978-682-2100

Ada and CORBA

GNACK and ORBit

From: okellogg@freenet.de (Oliver Kellogg)
Date: 18 Feb 2002 10:40:24 -0800
Subject: [announce] GNU Ada CORBA Kit version 1.0 released
Newsgroups: comp.lang.ada

Coinciding with the release of GNACK version 1.0, the ORBit-Ada project is happy to take its new home at SourceForge:
http://orbitada.sourceforge.net

Besides numerous bug fixes, the main addition in GNACK 1.0 is FATFIFI, a generator for Formatted Ada Text I/O From IDL. FATFIFI is a general purpose tool that can also be used without a CORBA ORB.

Furthermore, it is now possible to use Ada tasking in ORBit programs. This has been achieved by ORBit-GT, a version of ORBit-MT:
http://orbit-mt.sourceforge.net
that uses GNAT threads.

Both GNACK and ORBit-GT are available in the downloads section at http://sourceforge.net/projects/orbitada/

Ada and Microsoft

AdaGIDE 6.52 - Ada GUI IDE for Windows

From: "Gautier Write-only-address" <gautier_niouzes@hotmail.com>
Date: Sat, 30 Mar 2002 07:23:43 +0000
Subject: AdaGIDE 6.52 release (Ada GUI IDE for Windows 9x/NT/...)
Newsgroups: comp.lang.ada

Hi - just some news about latest release of AdaGIDE.

* Main improvements in AdaGIDE 6.52 compared to version 6.43.1

User definable colours and colour schemes

Parenthesis matching

Auto reformat on enter (can be disabled in Tools/Options)

Jumps to first error

Navigation among errors and warnings

Better working "Goto declaration" (Ctrl-G) XRef function

URL: http://www.usafa.af.mil/dfcs/bios/mcc_html/adagide.html

See sources and "About" box for contributions and contributors. Enjoy!

* Main improvements in AdaGIDE 6.43.1 compared to version 6.26 which comes with GNAT 3.13p:

Spell-checker

Automatic suggestion of filename on save

Support for multiple debuggers

Drag-and-drop for files

Updated options dialog

Gautier
http://www.mysunrise.ch/users/gdm/index.htm#Ada

CLAW now under GMGPL

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Mon, 22 Apr 2002 16:46:29 -0500
Subject: ANN: Claw Introductory Version Bindings now licensed under the GMGPL
Newsgroups: comp.lang.ada

Various people have suggested that the Claw Introductory Version would be more useful if it was licensed under a variant of the GPL. After lengthy deliberation, we have decided to make the Claw Introductory Version Bindings available under the GMGPL.

It will be a while before the download area of our website will be updated to reflect this change, most likely not until a new Introductory Version is released. Rest assured that we will enforce the license as if it is the GMGPL.

The license for other Editions of Claw, and for all versions of Claw GUI Builder, are unchanged for now. Thank you for your support of Ada.

Randy Brukardt President, R.R. Software, Inc

From: "David Botton" <David@Botton.com>
Date: Sun, 28 Apr 2002 01:03:05 -0400
Subject: ANN: GMGPL Claw Page
Newsgroups: comp.lang.ada

I've added a new page to AdaPower called the GMGPL Claw page.
http://www.adapower.com/claw
(AdaPower has and always will be an open forum for all Ada projects

regardless of my own projects and views.)

The GClaw project page is to create a central place for community work on the GMGPL version of Claw and to post various additions, examples and tools for all versions of CLAW

There are a number of examples for using Claw that were previous sent in by Tom Moran

I hope to put up some additions of my own to CLAW tomorrow.

I encourage others to submit examples, patches, and tools they may have written or will write in the future.

(No, this does not mean I am giving up GWindows. In fact there is even a new beta version up and I should have a number of updates ready within the week:-)

David Botton

References to Publications

Crosstalk articles

From: Richard Riehle <richard@adaworks.com>
Date: Wed, 06 Feb 2002 23:18:25 -0800
Subject: Ada Article in Crosstalk
Newsgroups: comp.lang.ada

There is an excellent article about a project completed in Ada in the February 2002 issue of Crosstalk. The web address is: http://www.stsc.hill.af.mil/crosstalk/crosstalk.html

[or maybe http://www.stsc.hill.af.mil/CrossTalk/2002/feb/feb02ind.asp -- jb]

The article is on page 25, titled:

"U.S. Army Develops High Quality, Extremely Low Cost Digital Message Parser" by Edgar Dalrymple.

Richard Riehle

From: Nielson Mark <Mark.Nielson@HILL.af.mil>
Date: Tue, 26 Feb 2002 16:51:09 -0700
Subject: The March 2002 Issue of CrossTalk is now available on-line.

The March 2002 issue of CrossTalk, The Journal of Defense Software Engineering is now available on our Web site at: http://www.stsc.hill.af.mil
Our theme this month is "Software by Numbers." We all know examples of real numbers from actual projects on their processes, quality, and return on investment are highly sought after but seldom seen. So in this issue we delve as deeply as we can into rounding up numbers and examples to share with you.

We begin by sharing real numbers from historical information collected by Donald J. Reifer from his collection of numerous projects. In Let the Numbers Do the Talking, he provides software cost and productivity benchmarks that you can use to determine how your organization ranks compared with industry averages. You will also be able to use his benchmarks to determine whether your software estimates are reasonable. Most importantly, he tells how not to abuse his information Our theme articles continue with How CMM Impacts Quality, Productivity, Rework, and the Bottom Line. Authors Michael Diaz and Jeff King share actual process improvement numbers from their company that are very useful for providing justification for process improvement and potential return on investment. Next, Walt Lipke shares one way his organization uses and benefits from their numbers in his article, Statistical Process Control of Project Performance.

As mentioned last month, we carried over until this issue Suzanne Garcia's article, Are You Prepared for CMMI? She talks about how applying technology adoption concepts can smooth the CMMI adoption process considerably.

Next, learn how one avionics project was able to achieve four-fold productivity and 10-fold quality improvements by adopting unambiguous programming languages that focus on preventing bugs in our article Correctness by Construction: Better Can Also be Cheaper by Peter Amey.

We end with an article by Frank Richey, Modeling and Simulation CMMI: A Conceptual View, which proposes enhancing the Capability Maturity Model Integration to include guidance for modeling and simulation.

I hope you enjoy our March 2002 issue on "Software by Numbers" and take away some actual basis for comparing and making improvements to your own software development processes.

Tracy L. Stauder Publisher

Contact the STSC Customer Service if you have any questions regarding your CrossTalk subscription or for additional STSC information: Software Technology Support Center Ogden ALC/TISE Attn: Customer Service 7278 4th Street Hill AFB, Utah 84056-5205

E-mail: karen.rasmussen@hill.af.mil
Voice: 801-775-5555, DSN 775-5555
Fax: 801-777-8069, DSN 777-8069

From: rod@praxis-cs.co.uk (Rod Chapman)
Date: 11 Mar 2002 08:38:19 -0800
Subject: (Another) Ada success story in CrossTalk magazine
Newsgroups: comp.lang.ada

There's a good Ada (and SPARK...) success story in the March 2002 issue of CrossTalk magazine - the title is "Correctness by Construction: Better can also be Cheaper" by Peter Amey of Praxis Critical Systems.

PDF is available from either the CrossTalk website (www.stsc.hill.af.mil) or from www.sparkada.com
- Rod Chapman, SPARK Team, Praxis Critical Systems

Abstract

For safety and mission critical systems,verification and validation activities frequently dominate development costs,accounting for as much as 80 percent in some cases. There is now compelling evidence that development methods that focus on bug prevention rather than bug detection can both raise quality and save time and money. A recent, large avionics project reported a four-fold productivity and 10-fold quality improvement by adopting such methods. A key ingredient of correctness by construction is the use of unambiguous programming languages that allow rigorous analysis very early in the development process.

Consolidated Ada Reference Manual

From: "Technical Webmaster" <Webmaster@adaic.com>
Date: Fri, 1 Mar 2002 00:19:22 -0600
Subject: [AdaIC] Consolidated Ada Reference Manual now available from Springer-Verlag

Springer-Verlag has published the Consolidated Ada Reference Manual. The new book merges the Ada 95 Reference Manual and the corrections and clarifications in the Technical Corrigendum that are now also part of the Ada standard. Though not an ISO publication, the Consolidated Ada LRM reflects a best effort to merge the two ISO documents.

The Consolidated Ada Reference Manual can be ordered directly from Springer-Verlag at
http://www.springerny.com/detail.tpl?isbn=3540430385
or at
http://www.springer.de/cgi-bin/search_book.pl?isbn=3-540-63143-7",
or from fine booksellers.

For more information, see http://www.adaic.org/news/cons-lrm.html.

To download a digital copy of the Consolidated Ada Reference Manual, see the Accessing the Ada Reference Manual page at http://www.adaic.org/standards/articles/lrm.html.

Ada Books Online

From: "Technical Webmaster" <Webmaster@adaic.com>
Date: Thu, 21 Feb 2002 20:09:32 -0600

Subject: [AdaIC] New versions of two on-line books have been posted.

New versions of two on-line Ada 95 books have been posted at the AdaIC.

A new version of Ada Distilled by Richard Riehle has been posted. This version corrects errors pointed out by many reviewers, and adds some new material. The book is aimed at experienced programmers who want to learn Ada at the programming level. Find "Ada Distilled" at: http://www.adaic.org/docs/distilled/adadistilled.pdf.

A minor update to John English's book, Ada95: The Craft of Object Oriented Programming has been posted. This version has minor corrections: the British usage of "full stop" instead of "period" confused some American readers, so "semicolon" has been substituted throughout to keep things language-independent. To browse the book, look at http://www.adaic.org/docs/craft/html/contents.htm.

To see all of the on-line textbooks see: http://www.adaic.org/free/freebook.html.

Randy Brukardt, Technical Webmaster http://www.adaic.org

A New Book in French

From: "Jean-Pierre Rosen" <rosen@adalog.fr>
Date: Thu, 2 May 2002 18:39:03 +0200
Subject: Nouveau livre
Newsgroups: fr.comp.lang.ada

La fine équipe de l'EIVD (programmation séquentielle/concurrente avec Ada 95) vient de sortir un nouveau livre: "Algorithmes et structures de données avec Ada, C++ et Java".

On y retrouve la même clarté et le même soin pédagogique que dans les précédents ouvrages. Les techniques classiques de tri, listes, graphes, tables et arbres y sont expliqués, avec implémentation dans les trois langages titre. A noter que les implémentations sont suffisemment proches pour permettre de comparer les langages, mais pas nécessairement identiques (on utilise plus l'héritage en Java qu'en Ada, par exemple).

Un excellent ouvrage de référence. En plus, on peut le donner à des programmeurs C++ pour leur faire découvrir Ada "par la bande"... :-)

J-P. Rosen (rosen@adalog.fr)

DDC-I Online News

From: JC <jcdk@ddci.com>
Date: Thu, 28 Feb 2002 12:40:19 -0700
Subject: Real-Time Industry Updates - News from DDC-I

DDC-I Online News February 2002, Volume 3, Number 2 - A monthly news update dedicated to DDC-I customers & registered subscribers.

This Month:

* Third Party Update: FAA DO-178B Training from Enea TekSci Learn more about Enea TekSci's FAA DO-178B training. Mention DDC-I and receive 50% off the next 2 day course, scheduled for April 18-19, 2002.

* On the Front Lines Meet Thomas E. Hansen, DDC-I A/S Support Manager

* Run and Debug Your Embedded 80x86/Pentium Code on a Plain PC in Real-Time Explore how this tool can reduce risk and offer substantial savings to your next development project.

* Customer Interaction and Software Development This article shares proven techniques including the link between customer interaction and the software development process.

* Tech Talk: Using Ada Library Information at the UNIX Command-line This tech note describes how the SCORE command-line tools for extracting Ada library information can be used for various tasks.

[...]

For the complete newsletter, go to http://www.ddci.com/news_vol3num2.shtml

From: JC <jcdk@ddci.com>
Date: Thu, 28 Mar 2002 16:05:09 -0700
Subject: Real-Time Industry Updates - News from DDC-I

DDC-I Online News March 2002, Volume 3, Number 3 - A monthly news update dedicated to DDC-I customers & registered subscribers.

This Month:

* Software Reuse This article discusses how reusing software saves you money by avoiding costly rewrites.

* On the Front Lines Meet Richard Frost - DDC-I, Inc. Assistant DACS Product Champion

* Third Party Update: ARTiSAN brings UML to the Ada Community Learn more about ARTiSAN's Ada code generator and their commitment to safety related projects.

* Tech Talk: Debugging SCORE UCC's on PowerPC This tech note offers proven time saving tips on debugging UCC's http://www.ddci.com/news_vol3num3.

* Introducing New Technology in the Workplace Implementing change in the workplace can be challenging. This article offers tips on how to get things going.

For the complete newsletter, go to http://www.ddci.com/news_vol3num3.shtml

From: JC <jcdk@ddci.com>
Date: Wed, 24 Apr 2002 12:13:11 -0700
Subject: Real-Time Industry Updates - News from DDC-I
DDC-I Online News April 2002, Volume 3, Number 4 - A monthly news update dedicated to DDC-I customers & registered subscribers.

This Month:

* Leaping out of Legacy Land Explore modernization upgrade options for your legacy program

* On the Front Lines Meet Thorkil Rasmussen - DDC-I A/S DACS Product Champion and valued employee for over 20 years!

* Third Party Update: AdaPower.com Ada developer resources & tools - A valuable resource for all Ada developers.

* Tech Talk: Handling of Normal and Fast Interrupts This note outlines features in DACS-80x86 systems and offers proven "good practice" ideas to ensure success.

* Lean Construction Every metaphor has it's limitations... an interesting comparison of software development to the construction industry.

For the complete newsletter, go to http://www.ddci.com/news_vol3num4.shtml

Java

Ada on the Palm

From: dewar@gnat.com (Robert Dewar)
Date: 22 Mar 2002 19:28:58 -0800
Subject: Re: Ada for Palm ?
Newsgroups: comp.lang.ada

> Is there an Ada for PalmOS handheld devices? I just like programmming while on the go (even in the swimming pool!) :-)

One approach is to fiddle with a JVM and JGNAT on the palm. We actually got an Ada program working on the Palm using this approach, but it was just a demo for fun, we did not follow it up further.

From: "D De Villiers" <~ddevilliers99@lando.co.za>
Date: Sun, 24 Mar 2002 17:20:13 +0200
Subject: Re: Ada for Palm ?
Newsgroups: comp.lang.ada

> I think there's a JVM for the Palm, so an Ada-to-Jcode compiler might be an option.

Yes! There is a JVM for Palm :) Visit www.palmos.com for info. on PalmOS development.

Lennie De Villiers

From: Jacob Sparre Andersen <sparre@nbi.dk>Date: Sat, 27 Apr 2002 16:16:31 +0200
Subject: Re: Ada for PalmOS?
Newsgroups: comp.lang.ada

> I have a program written in Ada that I would like to port to palmOS. Are there any compilers available for this? I don't think there are, so does anyone have suggestions how I might get this into a compiler for C/C++? I've never considered this kind of situation, so I'm not really sure where to go with it. If anyone has suggestions, I would really appreciate it!

GCC 3 can as far as I know be used as a cross-compiler targeting PalmOS, so it is "just" a matter of including the Ada front-end, when you compile GCC 3 in this configuration. I think you will have to manage without the GNAT run-time (no tasking) and use the PalmOS libraries for I/O.

Jacob

Ada Inside

Envisat1 Launched OK

From: "Jean-Pierre Rosen" <rosen@adalog.fr>
Date: Fri, 1 Mar 2002 15:46:36 +0100
Subject: Ada rocket launches Ada satellite
Newsgroups: comp.lang.ada

Tonight, Ariane 511 (whose flight control systems are in Ada) launched the Envisat1 satellite (with lots of Ada inside), a huge environmental survey satellite. This was a real challenge:

- Envisat1 is the biggest satellite ever built in Europe, and the biggest environment survey satellite ever launched: 10m long (26m deployed), for a weight of 8211Kg)
- Since it is a sun-synchronous satellite, the launch window was 0 (i.e. it had to be launched at the exact second).
- It was the first time an Ariane5 launched a polar satellite (i.e. towards north instead of East).
- It was the first time the long cap was used
- and more....

Thanks to new fuel-saving algorithms, it was possible to turn off the engine when the exact orbit was reached (with more than 3 seconds remaining fuel - quite a lot for such a rocket). Here is the result:

Requirement vs Achieved:

Req: Altitude: 7152.4km ± 7.5km.
Ach: 7152.4km

Req: Orbital position: 70.3 degrees ± 1.9.
Ach: 69.8

Req: Inclination: 98.5 degrees ± 1.1.
Ach: 98.5

All details available from http://www.arianespace.com/

-- J-P. Rosen (rosen@adalog.fr)

Subject: ARA News Release: Ada Flies Envisat Accurately
URL: http://www.adaic.org/news/envisat.html

BURLINGTON, MASS (MARCH 8, 2002)

The Ada Resource Association (ARA) announced another Ada success story today with the Ariane 511's launch of the Envisat 1 within the one-second tolerance allowed for the Sun-synchronized spacecraft. As the biggest satellite ever built in Europe, Envisat 1 also boasts being the largest environment survey satellite ever launched.

"The extraordinary accuracy of the Ariane's launch once again validates Ada for realtime use," said S. Tucker Taft, President of the ARA.

With flight control systems in Ada, the Ariane 511 for the first time succeeded in launching a "polar" satellite; i.e., towards the North instead of the traditional easterly direction. The Envisat 1 also depends on many Ada realtime software subsystems, including the radar altimeter.

The Envisat mission plan set a series of very specific and narrow windows for the launch and orbit. For example, the time for the Kourou, French Guiana, launch was 22:07 and 59 seconds, and Ada made sure that Flight 145 lifted off at one second before 22:08. Also, fuel-saving algorithms in Ada allowed the engine to turn off when the Envisat 1 reached its precise orbit three seconds before the scheduled time. It began orbiting within the right 100 meters where tolerance was 7.5 kilometers. As a result, the 26-meter long (when deployed), 8211-kilogram rocket carries a minimum of fuel.

For the next five years, the satellite will record data on how humans are affecting Earth's health, including through patterns of land use and its effect on soil moisture and fertility; the quantity and size of ice flows; and the amount of ozone and other chemicals in the atmosphere observed on the planet's "limb" or edge.

Developed in a European Space Agency program, the satellite will be operated from ESA's European Space Operations Center (ESOC) in Darmstadt, Germany. Europe's Astrium led an industry consortium of 50 companies to produce the Envisat spacecraft.

For more information on Ada and the Envisat, please write to Ann Brandon, Communications Director Ada Resource Assoc. abrandon@sover.net

Ada Survey

From: clanfear@yahoo.com (cal)
Date: 6 Feb 2002 09:51:31 -0800
Subject: Ada research survey
Newsgroups: comp.lang.ada

VDC is conducting the first comprehensive market study of the Ada language. As part of this research we are looking for developers to complete a short survey about their experiences with Ada. Your opinions will help tools vendors to design better tools and understand your needs.

There will be a prize drawing at the end of the research.

The survey is at http://www.vdc-corp.com/testada

> Do you mean here that the survey data will be public?

The survey data will be used in research report which is being subscribe to by leading tools vendors. VDC regularly publishes white papers on top embedded websites and in print publications that contain research highlights.

I would even publish the white paper here if enough folks helped us out. The white paper would be of interest to many people on this list.

Ada is an under covered topic and the more input the better.

The folks on this list care about Ada so help me tell the tools vendors what you want and need.

Thanks

Ada in Context

On Buffer Overflow

From: Nick Roberts <nickroberts@ADAOS.WORLDONLINE.CO.UK>
Date: Wed, 6 Feb 2002 18:29:06 -0000
Subject: Re: Buffer Overflow Propensity as a Function of Programming Language
To: TEAM-ADA@ACM.ORG

"Thomas A. Panfil" <t.panfil@gte.net> wrote:

> Hi All, I'd like to be able to cite a good paper on why Buffer Overflow susceptibility is common in software written in some popular language(s), and rare or relatively easy to prevent when using other languages. Advice, anyone?

Some hopefully relevant points.

For a buffer overflow vulnerability to be actually exploitable, it is necessary for:

(a) the underlying operating system or execution environment to fail to provide or deploy protection against the execution of code that lies in an area of memory which is read-write [1];

(b) the underlying operating system or networking software configuration to fail to isolate the executional environment of the (TCP) service application to the maximum extent feasible [2];

(c) the service application to have a twofold combined kind of bug, whereby the client is able to write binary data into a certain area of the service application's memory, and then cause somehow the service application to start the execution of instructions somewhere within that area of memory [3].

[1] The C or C++ language often prevents the use of such protection, even when it is available (at no executional cost) on the architecture. While some forms of protection can be used, others cannot (because of C's need for a 'flat' address space). Ada does not require a flat address space (but typically suffers from the limitation of having to interface to C software to be able to use operating system specific functions, of course).

[2] This especially pertains to running the service application as a normal user (rather than root), and ensuring that user has the minimal (file) permissions necessary to do its job. A typical situation on most UNIX-based operating systems, unless the system administrator is very sophisticated, and an even more typical situation on Windows NT (with IIS), is that such elementary precautions are not taken.

[3] This is a theoretically extremely unlikely bug, that nevertheless demonstrably tends to crop up within (large) C and C++ software, and to my knowledge never in software written in any other language.

The thing that is most deadly about a successful buffer overrun exploit attack (and similar types of attack), is that the attacker gets to run his own code, often with root privileges, and can thus truly "do anything he likes" from that point on. Typically the host computer is totally compromised; this in turn (if the attacker is skilled and persistent) can lead to whole networks being compromised.

I believe there is a negligible likelihood of a TCP or UDP service application written in Ada, especially with most or all checks left on, suffering from a buffer overrun vulnerability, or any vulnerability that permits the client to cause it to execute arbitrary code. This is regardless of the compiler, the host machine (architecture), and the host operating system. (Note however that this is not to be confused with the case of an Ada main program using substantial library code written in C.)

A good starting point for more information may be the US DoE Computer Incident Advisory Capacity (CIAC) at: http://www.ciac.org/ciac

I believe that with a bit of digging you will find much fodder for your research. Happy hunting!

-- Nick Roberts

From: "John McCormick" <mccormic@CS.UNI.EDU>
Date: Wednesday, February 06, 2002 10:11
Subject: Re: Buffer Overflow Propensity as a Function of Programming Language

I recall from my analysis of the data used by Mark Eisenstadt to classify really difficult bugs that there were a good number of bugs resulting from array bounds errors. Errors that Ada and Java would have caught.

Mark's article is Eisenstadt, M. (1997). My Hairiest Bug War Stories. Communications of the ACM, 40, 30-37.

While he didn't give all the data on the bugs, he will make it available on request.

John W. McCormick, Computer Science Department, University of Northern Iowa, Cedar Falls, IA 50614-0507, http://www.cs.uni.edu/~mccormic/

From: Roger Gariépy <rgariepy@ROCLER.QC.CA>
Subject: Re: Buffer Overflow Propensity as a Function of Programming Language
You can find a link to the paper of M. Eisenstadt at: http://www.adahome.com/articles/1997-05/am_bugs.html

Roger Gariépy, rgariepy@rocler.qc.ca

In and out of Oblivion

From: Volkert.Barr@freenet.de (Volkert)
Date: 30 Jan 2002 15:09:04 -0800
Subject: Ada's Slide To Oblivion ...
Newsgroups: comp.lang.ada

Found at Embedded Systems Programming:

>Ada is the only language designed to significantly reduce and maybe even eliminate dumb programming errors. Did it fall into disuse because we're intellectually lazy?

read more:
http://www.embedded.com/story/OEG20020125S0098

Volkert

From: "Marin David Condic" <dont.bother.mcondic.auntie.spam@acm.org>
Date: Wed, 30 Jan 2002 18:57:41 -0500
Subject: Re: Ada's Slide To Oblivion ...
Newsgroups: comp.lang.ada

An interesting article. One could argue about the accuracy of the survey, but it probably isn't that far off from reality.

What I liked about it was that it was fair and balanced. It didn't smack of the usual anti-Ada vitriol, nor was it filled with misinformation. The criticism that Ada doesn't have as many tools as C/C++ is reasonably fair - I think it is a better situation than the author seems to imply, but let's face it: For just about any embedded board, you can get a C compiler thrown in with the development kit & you won't find Ada riding along with it as an alternate choice. (Although Gnat merging with gcc stands to help improve the situation - but still people have to ask for it or nobody will bother.)

The question about programmers being "intellectually lazy" may have a lot to do with it. In order to do Ada development in a way that maximizes the benefits and minimizes the time fighting with the compiler to get it right, requires that you spend time up front thinking about the organization of the system - what the relevant data types are, what information should be hidden, etc. Embedded developers tend to be tinkerers who want to start hacking some bootstrap code and keep adding things to it until it works. Weeks of planning and diagram drawing and design meetings prior to writing any code tends to not be the thing they got into the business to do. Never mind that it might save months/years of debugging and produce a more reliable system that improved customer satisfaction and reduced liability - that's just not the mode of thought that feels comfortable to your average embedded/C developer.

The question at the end about the government being to blame for not sticking to its guns is another interesting one. The government instituting "The Mandate" (especially when compiler technology just wasn't there) probably raised a lot of hackles over being "forced" to do something. (I *still* think that had the government tried bribery instead of extortion, it might have worked. If you were the program manager for some electronic whozits and the government offered you a $100,000.00 bonus if only you could find a way to get the project done in Ada, do you think your opposition to Ada would be so strong?)

Anyway, having had The Mandate, then abandoning it is worse than never having The Mandate to begin with. Think about it - the perception is that the government was admitting it made a mistake by mandating Ada, so the contractors started abandoning it in droves. Standing there saying "No! Really! I'm *NOT* saying Ada is a bad thing!!!!" doesn't matter. Actions speak louder than words and perception often *IS* reality. ("Hey, the DoD dropped Ada like a hot rock. We must have been right all along. Ada really *did* suck!)

The good news is that if people are writing thoughtful articles like this and observing that Ada really does have benefits (despite lack of use) maybe it might generate some renewed interest. The fact that they're writing about it at all is a sign that Ada isn't a non-issue. IOW, "I don't care what they say about Ada as long as they capitalize its name right!" :-)

Marin David Condic, Senior Software Engineer, Pace Micro Technology Americas, www.pacemicro.com

From: Jim Rogers <jimmaureenrogers@worldnet.att.net>
Date: Thu, 31 Jan 2002 02:37:52 GMT
Subject: Re: Ada's Slide To Oblivion ...
Newsgroups: comp.lang.ada

Comments and articles similar to this appear occasionally.

I like to try to read between the lines and understand the assumptions being made by the author. In the case of this article I find one assumption is that people know Ada as well as C, and have made a conscious decision toward C and away from Ada. I do not believe this assumption is even approximately true.

Most of the embedded programmers currently working embedded software engineers have only heard rumors about Ada. Most have never used it. Many have never seen it or heard about it.

C++ has grown because C programmers were convinced it was really C with a few unimportant differences. In other words, C++ was designed to fool people into adopting it. The same cannot be said about Ada. My contention is that Ada has never slid into oblivion. In fact, Ada is slowly climbing out of the initial oblivion into which it was born.

Jim Rogers, Colorado Springs, Colorado, USA

Conference Calendar

This is a list of European and large world-wide events that may be of interest to the Ada community. More information on items marked ♦ is available elsewhere in the *Journal*. The information here is extracted from the online *Conference announcements for the international Ada community* at http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium webserver. These pages contain full announcements, calls for papers, calls for participation, programmes, URLs etc and are updated regularly.

2002

07-10 August	**Metainformatics Symposium 2002** Esbjerg, Denmark. Topics include: software engineering, object-oriented programming, design patterns, component-based systems, middleware, programming environments, programming languages, development environments, etc.
18-21 August	**2002 International Conference on Parallel Processing (ICPP'02)** Vancouver, British Columbia, Canada. Topics include: Programming Methodologies and Tools; Compilers and Languages; Parallel/Distributed Algorithms; etc.
18-21 August	**3rd International Workshop on Metacomputing Systems and Applications (MSA'2002)** Topics include: Programming Models; Programming Languages; etc.
18-22 August	**2002 Rational Software User Conference (RUC'2002)** Lake Buena Vista, Florida, USA. Topics include: case studies featuring one or more Rational products, provide practical tips and techniques geared towards intermediate or advanced users, etc.
19-22 August	**2nd Software Product Line Conference (SPLC2)** San Diego, California, USA
20-23 August	**13th International Conference on Concurrency Theory (CONCUR'2002)** Brno, Czech. Repubic. Topics include: concurrency related aspects of: real-time systems, distributed programming, object-oriented programming, case studies, tools and environments for programming and verification, etc.
25-30 August	**3rd Working IEEE/IFIP Conference on Software Architecture (WICSA'2002)** Montreal, Canada Co-located with 17th IFIP World Computer Conference (WCC)
26-28 August	**International Conference on Pervasive Computing (PERVASIVE'2002)** Zurich, Switzerland.
26-29 August	**26th Annual International Computer Software and Applications Conference (COMPSAC'2002)** Oxford, England. Theme: Prolonging Software Life: Development and Redevelopment. Topics include: Component-based; Object-oriented technology; Quality management; Safety and security; Software architecture, software development framework, and design; Software evolution; Software fault tolerance; Software re-engineering; Software reliability; Software reuse; Distributed systems; Embedded systems; Enterprise systems; Middleware systems; etc.
27-30 August	**European conference on Parallel Processing (Euro-Par'2002)** Paderborn, Germany. Topics include: Support Tools and Environments; Performance Evaluation, Analysis and Optimization; Distributed Systems and Algorithms; Parallel Programming: Models, Methods and Programming Languages; etc.
02-05 September	**8th International Conference on Object-Oriented Information Systems (OOIS'2002)** Montpellier, France. Topics: OO frameworks; OO components/COTS; OO patterns; OO middle-ware; Reuse processes; Web-based Applications; OO distributed systems; OO built-in tests; etc. Includes a.o. the following events:
	02 September **OOIS2002 - Workshop on MAnaging of SPEcialization/Generalization HIerarchies (MASPEGHI)** Deadline for paper submissions: May 6, 2002.
	02 September **OOIS2002 - Workshop on the planning and management of organisational transition to Object Technology**
	02 September **Workshop on Reuse in Object-Oriented Information Systems Design**

03-06 September	**International Internet & Software Quality Week 2002 (QW'2002)** San Francisco, California, USA Theme: "The Wired World..." Topics: Application of Formal Methods; Software Reliability Studies; Object Oriented Testing; Productivity and Quality Issues; Real-Time Software; Real-World Experience; etc.
03-06 September	**IEEE Symposia on Human-Centric Computing Languages and Environments (HCC'02)** Arlington, VA, USA Topics: Design, formalization, implementation, and evaluation of computing languages that are easier to learn, easier to use, and easier to understand by a broader group of people.
04-06 September	**3rd International Conference on Parallel and Distributed Computing, Applications, and Techniques (PDCAT'2002)** Kanazawa, Japan. Topics include: Formal methods and programming languages; Parallelizing compilers; Web technologies; Component-based and OO Technology; Tools and environments for software development; etc.
04-06 September	**EUROMICRO Conference** Dortmund, Germany.
	04-06 September **EUROMICRO Conference Track on Software Process and Product Improvement**
	04-06 September **EUROMICRO Conference Track on Component-based Software Engineering** Topics include: Components and Reuse; Component Specification; Component Design, Implementation, Testing; Development Environment and Tools; Components for Real-time systems; Component-based embedded systems; Case Studies; etc.
	04-06 September **EUROMICRO Conference Track on Work in Progress** Topics include: Software Process and Product Improvement; Component-based Software Engineering; etc.
09-10 September	**8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2002)** Essen, Germany
09-12 September	**7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'2002)** University of Oldenburg, Germany.
09-13: September	**9th International Conference on Algebraic Methodology And Software Technology (AMAST'2002)** St. Gilles les Bains, Reunion Island, France
09-13 September	**IEEE Joint Conference on Requirements Engineering (RE'02)** Essen, Germany.
09-13 September	**3rd Argentine Symposium in Software Engineering (ASSE'2002)** Santa Fe, Argentina Topics include: Software Quality; Distributed Objects; Reuse and Components; Design Patterns; Practice of object-oriented technology and its application in industrial environments; Education in software engineering; etc.
09-13 September	**International Conference on Practical Software Quality Techniques & Testing Techniques (PSQT/PSTT'2002 North)** St. Paul, Minnesota, USA.
10-13 September	**21st International Conference on Computer Safety, Reliability and Security (Safecomp'2002)** Catania, Italy. Focuses on safety-critical computer applications.
15-18: September	**Conference on Communicating Process Architectures 2002 (CPA'2002)** Reading, UK Topics include: concurrent design patterns and tools; safety and security issues (race-hazards, deadlock, livelock, process starvation, ...); language issues; applications: scientific (including graphics and GUIs), engineering (including embedded, real-time and safety-critical), business (including mobile and e-commerce) and home (including entertainment); etc.
17-20 September	**9th International Static Analysis Symposium (SAS'2002)** Madrid, Spain Topics include: abstract interpretation; data flow analysis; verification systems; optimizing compilers; etc.
17-20 September	**6th International Enterprise Distributed Object Computing Conference (EDOC'2002)** Lausanne, Switzerland.
18-20 September	**European Software Process Improvement Conference (EuroSPI'2002)** Nuremberg, Germany

23-27 September	**17th IEEE International Conference on Automated Software Engineering (ASE'2002)** Edinburgh, U.K.
24-27 September	**Forum on Specification and Design Languages (FDL'2002)** Marseille, France
24-27 September	**8th IEEE Real-Time Technology and Applications Symposium (RTAS'2002)** San Jose, California, USA. Topics include: Real-time applications in Linux; Real-time software components; Embedded control applications; Secure real-time systems; Middleware support; etc. Includes:
	24 September **Workshop on Embedded Systems Codesign (ESCODES'2002)**
29 September – 2 Oct.	**4th Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS'2002)** Linz, Austria. Topics include: Parallel and Distributed Algorithms; Languages, Tools and Environments; Applications; Distributed OO Systems; Middlewares; etc.
30 September – 04 Oct.	**5th International Conference on UML - the Language and its Applications (UML'2002)** Dresden, Germany. Theme: "Model Engineering, Concepts and Tools"
01-04 October	**2002 IASTED International Conference on Networks, Parallel and Distributed Processing, and Applications (NPDPA'2002)** Tsukuba Science City, Japan Topics include: Distributed Processing, Distributed Real-time Systems, Parallel Processing, Parallel Programming, Parallel Computing Systems, Heterogeneous Computing, Compilers, Real Time and Embedded Systems, Applications, Fault Tolerance, Reusability, Reliability, etc.
02 October	**8th IEEE Workshop on Empirical Studies of Software Maintenance (WESS'2002)** Montreal, Quebec, Canada.
03-04 October	**2002 International Symposium on Empirical Software Engineering (ISESE'2002)** Nara, Japan Topics include: Evaluation of the readability of coding styles; Reports on the benefits derived from using software development environments; Development of predictive models of defect rates and reliability from real data; Industrial experience in process improvement; Quality measurement; Experience management; etc.
03-06 October	**IEEE International Conference on Software Maintenance (ICSM'2002)** Montreal, Canada. Theme: Maintaining distributed heterogeneous systems. Topics include: Design for maintenance; Formal methods; Software reusability; Empirical studies; Programming languages; Maintenance and/or productivity metrics; Preventive maintenance; Tools and environments; Freeware and open source applications; Internet and distributed systems; Source code analysis and manipulation; Impact of new software practices; etc. Includes:
03-08 October	**Principles, Logics, and Implementations of high-level programming languages (PLI'2002)** Pittsburgh, USA
06-08 October	**1st ACM SIGPLAN/SIGSOFT Conference on Generators and Components (GCSE/SAIG'2002)** Pittsburgh, PA, USA
06-10 October	**10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)** San Jose, California, USA. Topics include: Interaction of operating systems, compilers,. programming languages, and architectures; Case studies of hardware/software design in novel experimental systems; etc.
07-09 October	**2nd Workshop on Embedded Software (EMSOFT'02)** Grenoble, France. Topics include: System design and integration methodologies, Programming languages and software eng., etc.
08-11 October	**International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES'2002) Grenoble**, France Co-located with EMSOFT'2002 Topics include: Compilers and Operating Systems (New optimizing compilers for embedded-domain constraints, Compiler controlled memory hierarchy management and smart caches, ...); Architecture (Synergy between extant parallel computing technologies, such as notations for expressing concurrency, and instruction level parallel processing, ...); Tools and Methodologies (Automated design and synthesis of application- or domain-specific processors, ...); Applications; etc.
13-16 October	**21st Symposium on Reliable Distributed Systems (SRDS'2002)** Osaka University, Suita, Japan. Topics include: Distributed systems with reliability, availability, security, safety, and/or

	real-time requirements; Distributed databases and transaction processing; Distributed objects and middleware systems; Security and high confidence systems; Analytical or experimental evaluations of reliable distributed systems; etc.
	13 October **SRDS2002 - International Workshop on Self-Repairing and Self-Configurable Distributed Systems (RCDS'2002)**
16-18 October	**16th Brazilian Symposium on Software Engineering (SBES'2002)** Gramado, Brazil Topics include: Industrial applications of Software Engineering; Component-based Software Engineering; Theoretical Foundations of Software Engineering: Formal specification, refinement, software validation and verification; Methods, Techniques, Languages and Tools for Software Engineering; Software Maintenance; Software Quality; Software Reuse; Software verification, validation and testing; etc.
♦ 17 October	**Combined Ada UK / Embedded Systems Club Conference** UK. Topics include: any topic relevant to the embedded systems and/or Ada communities.
18-20 October	**Conference on Quality Engineering in Software Technology (CONQUEST'2002)** Nuremberg, Germany.
22-25 October	**4th International Conference on Formal Engineering Methods (ICFEM'2002)** Shanghai, China
27-31 October	**21st Digital Avionics Systems Conference (DASC'2002)** Irvine, California, USA. Topics include: avionics (flight critical systems, system engineering, open systems, software engineering, etc.), Air Traffic Management, etc.
28-30 October	**12th International Conference on Software Quality (ICSQ'2002)** Ottawa, Ontario, Canada
28 October – 01 Nov	**9th IEEE Working Conference on Reverse Engineering (WCRE'2002)** Richmond, Virginia, USA Topics include: Software maintenance and evolution; Program comprehension; Software architecture extraction; Software migrations; Transitioning to product lines; Experience reports; Preprocessing and parsing; Software components; Reverse engineering tool support; UML and round trip engineering; Software metrics; etc.
28 October – 01 Nov.	**4th International Symposium on Distributed Objects and Applications (DOA'2002)** Irvine, California, USA. Topics include: Design patterns for distributed object design; Interoperability-supporting environments; Security, including authorisation and authentication; Reliable and fault tolerant middlewares; Real-time/Reflective middlewares; Web Services and distributed objects, including SOAP interoperability and service discovery; Reports on Best Practice; etc. Deadline for paper submissions: May 31, 2002.
04-08 November	**17th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'2002)** Seattle, WA, USA. Deadline for submissions: July 19, 2002 (Posters, Demonstrations, Doctoral Symposium, and Student Volunteers).
11-14 Novermber	**International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'2002)** Houston, Texas, USA Theme: "Formal Methods for Protocol Engineering and Distributed Systems" Topics include: Formal approaches to concurrent/distributed Object-Oriented systems; Real-time and probability aspects; Verification and validation; Relations between informal and formal specification; Software tools and support environments; Practical experience and case studies; Corporate strategic and financial consequences of using formal methods; etc.
12-15 November	**13th International Symposium on Software Reliability Engineering (ISSRE'2002)** Annapolis, Maryland, USA. Topics include: Software testing and verification; Secure software engineering; Security testing and certification; Reliability of distributed systems; Standards and regulation; etc. Deadline for submissions: May 1, 2002 (tutorials, panels), August 1, 2002 (student papers, fast abstracts)
18-21 November	**1nternational Conference on Software Process Improvement (ICSPI'2002)** Washington DC Area, USA

18-22 November	**ACM SIGSOFT 2002 10th International Symposium on the Foundations of Software Engineering (FSE-10)** Charleston, South Carolina, USA. Topics include: Component-Based Software Engineering; Empirical Studies of Software Tools and Methods; Feature Interaction and Crosscutting Concerns; Generic Programming and Software Reuse; Software Engineering Tools and Environments; Software Reliability Engineering; Software Safety; Specification and Verification; etc. Deadline for submissions: August 15, 2002 (student posters).
02-04 December	**8th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'2002)** Greenbelt, Maryland, USA. Topics include: technologies for developing complex systems; means of avoiding, controlling, or coping with complexity; embedded real time complex systems; distributed and network based complex software systems; design and analysis of complex software systems; formal methods for complex systems; techniques for component-based software development; etc. Deadline for submissions: May 3, 2002 (initial abstracts), May 10, 2002 (papers, extended abstracts), June 7, 2002 (tutorials, panels and exhibits).
03-05 December	**15th International Conference on Software & Systems Engineering and their Applications (ICSSEA'2002)** Paris, France
04-06 December	**27th Annual Software Engineering Workshop (SEW27)** Greenbelt, MD, USA Co-located with ICECCS'2002 Topics include: Software quality assurance; Software engineering processes and process improvement; Real-time Software Engineering; Software maintenance, reuse, and legacy systems; etc.
04-06 December	**9th Asia-Pacific Software Engineering Conference (APSEC'2002)** Grand Mercure Broadbeach, Gold Coast, Queensland, Australia
08 December	**2nd Workshop on Industrial Experiences with Systems Software (WIESS'2002)** Boston, Massachusetts, USA. Topics include: Distributed Systems, Programming Environments and Tools, Fault Tolerance and High Availability, Real Time and Quality of Service, Middleware, Embedded Systems, etc.
♦ 08-12 December	**2002 ACM SIGAda Annual International Conference (SIGAda'2002)** Houston, Texas, USA. Topics include: Reliability needs and styles; Safety and high integrity issues; Use of the Ada Distributed Systems Annex; Process and quality metrics; Testing and validation; Standards; Use of ASIS for new Ada tool development; Relationships between Ada and real-timeJava; Mixed-language development; Ada in XML environments; Ada education; Use of Real-Time CORBA; Real-time networking/quality of service guarantees; Fault tolerance and recovery; Distributed system load balancing; Static and dynamic code analysis; Performance analysis; Debugging complex systems; Integrating COTS software components; System Architecture & Design..
09-11 December	**5th USENIX Symposium on Operating Systems Design and Implementation (OSDI'2002)** Boston, Massachusetts, USA. Topics include: distributed systems, parallel systems, embedded systems, the influence of hardware development on systems and vice-versa, etc.
09-11 December	**4th International Conference on Product Focused Software Process Improvement (PROFES'2002)** Rovaniemi (Arctic Circle), Finland Topics include: Software Quality; Methods and Tools; Industrial Experiences and Case Studies; Best practices; Lessons Learned; Embedded Systems; etc.
10 December	**Birthday of Lady Ada Lovelace, born in 1815 – Happy Programmers' Day!**
16-18 December	**2002 Pacific Rim International Symposium on Dependable Computing (PRDC'2002)** Tsukuba, Japan Topics include: Design for system dependability; Fault-tolerant systems and software; Fault tolerance for parallel and distributed systems; Software and hardware reliability, verification and testing; Tools for design and evaluation of dependable systems; Application-specific dependable system (e.g., embedded systems, WWW servers, transaction processing); etc**.**

2003

06-09 January	**Software Technology Track of the 36th Hawaii International Conference on System Sciences (HICSS-36)** Big Island of Hawaii, USA. Includes mini-tracks on: Experimental Software Engineering; Domain-Specific Languages; Distributed Object and Component-based Software

	Systems (Design Patterns for Distributed Systems, Middleware, Programming Languages and Environments for Distributed Object and Component Systems, ...); etc.
15-17 January	**30th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'2003)** New Orleans, Louisiana, USA Topics include: design, definition, analysis, and implementation of programming languages, programming systems, and programming abstractions.
03-05 February	**9th International Conference on Languages and Models with Objects (LMO'2003)** Vannes, France Topics include (in French): Programmation par objet et programmation par composant; Programmation par objet et modélisation par objets; etc. Deadline for submissions: September 2, 2002
04-07 February	**Australasian Computer Science Conference (ACSC'2002)** Adelaide, South Australia Topics include: Compilers, Concurrency, Distributed Systems, Embedded Systems, Fault Tolerance, Formal Methods, Object-Oriented Systems, Programming Languages, Real-time Systems, Reliabiliity, Software Engineering, Trusted Systems, etc. Deadline for submissions: September 6, 2002
05-07 February	**11th Euromicro Conference on Parallel Distributed and Network based Processing (PDP'2003)** Genoa, Italy. Topics include: Distributed Systems; Parallel Computer Systems; Models and Tools for Parallel Programming Environments; Advanced Applications (numerical applications with multi-level parallelism, real time distributed applications, distributed business applications, ...); Languages, Compilers and Runtime Support Systems (task and data parallel languages, object-oriented languages, scheduling and load balancing, task and object migration, ...), etc. Special sessions on: Advanced Tools for Parallel and Distributed Programming; Parallel Realtime Systems; etc.
10-12 February	**2nd International Conference on Commercial Off-The-Shelf (COTS)-Based Software Systems (ICCBSS'2003)** Ottawa, Canada Theme: "Multiple Paths, Multiple Solutions"
24-27 February	**15th Annual Software Engineering Process Group Conference (SEPG'2003)** Boston, Massachusetts, USA Theme: "Assuring Stability in a Global Enterprise" Deadline for submissions: Jul 01, 2002 (session proposals)
09-12 March	**2003 ACM Symposium on Applied Computing (SAC'03)** Melbourne, Florida, USA Includes tracks on Embedded Systems: Applications, Solutions, and Techniques; Software Engineering: Applications, Practices and Tools; etc. Deadline for submissions: April 15, 2002 (track proposals), September 6, 2002 (papers and tutorials)
20-22 March	**16th Conference on Software Engineering Education and Training (CSEET'2003)** Madrid, Spain Theme: "Software Engineering in Industry and University for the 21st Century" Deadline for submissions: October 1, 2002
26-28 March	**7th European Conference on Software Maintenance and Reengineering (CSMR'2003)** Benevento, Italy Topics include: experience reports, enabling technologies, etc. Deadline for submissions: October 10, 2002
05-13 April	**European Joint Conferences on Theory and Practice of Software (ETAPS'2003)** Warsaw, Poland. Event includes: conferences from 7 to 11 April 2003, affiliated workshops on 5-6 and 12-13 April, 2003.
03-10 May	**International Conference on Software Engineering (ICSE'2003)** Portland, Oregon, USA Deadline for submissions: October 4, 2002 (Software Engineering Education Track)
♦ 16-20 June	**8th International Conference on Reliable Software Technologies - Ada-Europe'2003** Toulouse, France Sponsored by Ada-Europe, in cooperation with ACM SIGAda (approval pending). Deadline for submissions: October 31, 2002 (papers, extended abstracts, tutorials, workshops)

Autumn Conference, 17th October 2002

Register Your Interest Now !

Attending as a delegate	θ	Giving a Tutorial	θ	Presenting a case study, vendor presentation or technical paper	θ
Chairing a session	θ	Submitting a delegate position paper	θ	Organising a workshop or "BOF"	θ

Name: ____________________ Position: ____________________

Company: ____

Address: ____

____________________ Post Code: ____________________ Country: ____________________

Tel: ____________________ Fax: ____________________

Email: ____

We invite contributions on any topic relevant to the embedded systems and/or Ada communities. We are particularly interested in receiving proposals for the following types of sessions:

- **Case studies**: presentations, typically 60 minutes in duration, reporting on experience of applying embedded technologies in real-world applications.
- **Tutorials**: training sessions, typically half a day or one day in duration, with the emphasis on equipping developers with new skills and techniques.
- **Workshops / Birds-of-a-feather sessions (BOFs):** BOFs give people with common interests the opportunity to engage in substantive discussions, sharing lessons learned and establishing relationships that may continue beyond the conference. Workshops are more formal and attendees are often chosen by submission of appropriate **position papers**.
- **Delegate position papers**: a position paper is usually around 2 to 5 pages, although it can be longer if the technical contribution demands it, in which you to present an opinion, viewpoint or experience relevant to the community. Position papers are ideal for delegates who wish to encourage the conference to address specific issues without the need to get up and speak to an audience.
- **Vendor presentations**: commercial presentations, typically 15 to 20 minutes in duration, describing new product releases or product enhancements of interest to the community.
- **Technical papers**: presentations, typically 60 minutes in duration, covering a technical topic related to embedded systems engineering.

If you are interested in submitting a paper or proposal for a session, please contact us immediately to register your interest. We will then contact you to agree a timetable for submission.

ADA uk
Ada UK User Group

Hazel Lawton,
The Embedded Systems Club/Ada UK,
Adaxia Ltd, PO Box 376,
Chesterfield S42 7YB, UK
Fax +44 (0) 1246 567339
Email: Hazel@Adaxia.com

The Embedded Systems Club

www.AdaUK.org.uk

www.EmbeddedSystemsClub.com

Preliminary Call for Participation – SIGAda 2002

8-12 December 2002, Houston, Texas, USA
Sponsored by ACM SIGAda
http://www.acm.org/sigada/conf/sigada2002
(Approval pending by ACM)

Constructing reliable software is an engineering challenge. The application of methods, tools, and languages interrelate to make the challenge easier or more difficult. This conference focuses on the interaction between these three aspects of software engineering, especially how features in a language such as Ada drive the tools, methods, and ultimately correctness, reliability, and quality of the resulting software. Especially welcome are papers that analyze Ada with respect to these factors or in comparison with other languages. This conference will gather industrial experts, educators, software engineers, and researchers interested in developing and testing reliable software. Technical or theoretical papers as well as experience reports with a focus on Ada are solicited. Possible topics include but are not limited to:

- Reliability needs and styles
- Safety and high integrity issues
- Use of the Ada Distributed Systems Annex
- Process and quality metrics
- Testing and validation
- Standards
- Use of ASIS for new Ada tool development
- Relationships between Ada and real-time Java
- Mixed-language development
- Ada in XML environments
- Ada education
- Use of Real-Time CORBA
- Real-time networking/quality of service guarantees
- Fault tolerance and recovery
- Distributed system load balancing
- Static and dynamic code analysis
- Performance analysis
- Debugging complex systems
- Integrating COTS software components
- System Architecture & Design

How You Can Contribute

SIGAda 2002 is interested in receiving contributions in six major categories. Contributions from students are actively solicited. **Technical Articles** present significant results in research, practice, or education. These papers will be double-blind refereed and published in the Conference Proceedings. Papers should not exceed 5000 words (equivalent to approximately 10 pages, typeset 10-point on 16-point spacing). **Extended Abstracts** discuss current work for which early submission of a full paper may be premature. If your abstract is accepted, you will be expected to produce a full paper, which will appear in the proceedings. Extended abstracts will be competitively reviewed. Clearly state the contribution of the work being described, its relationship with previous work by you and others (with bibliographic references), results to date, and future directions. Please do not exceed 2500 words (equivalent to approximately 5 pages typeset 10-point on 16-point spacing). **Experience Reports** present timely results on the application of Ada and related technologies to the design and implementation of applications such as the following: avionics, aerospace, automobile, command and control, consumer electronics, process control, transportation, trading systems, energy, medical systems, simulation, telecommunications, etc. Such reports will be selected on the basis of the interest of the experience presented to the community of Ada practitioners. You are invited to submit a 1-2 page description of the project and the key points of interest of project experiences. Descriptions will be published in the final program or proceedings, but a paper will not be required. **Workshops** are focused work sessions, which provide a forum for knowledgeable professionals to explore issues, exchange views, and perhaps produce a report on a particular subject. A list of planned workshops and requirements for participation will be published in the SIGAda 2002 Advance Program. Workshop proposals will be evaluated by the Program Committee and selected based on their applicability to the conference and potential for attracting participants. Proposals should state the problem or issue to be addressed, the coordinator(s), and criteria for participant selection. **Panel Sessions** gather a group of experts on a particular topic who present their views and then exchange views with each other and the audience. Panel proposals should be 1-2 pages in length, identifying the topic, coordinator, and potential panelists. **Tutorials** offer the flexibility to address a broad spectrum of topics relevant to Ada, and those enabling technologies which make the engineering of Ada applications more effective. Submissions will be evaluated based on relevance, suitability for presentation in tutorial format, presenter's expertise, and past performance. Tutorial proposals should include the expected level of experience of participants, an abstract or outline, the qualifications of the instructor(s), and the length of the tutorial.

Please submit **Technical Articles, Extended Abstracts, Experience Reports, Workshop proposals, and Panel Sessions** to the Program Chair, John McCormick <McCormick@cs.uni.edu> and **Tutorial proposals** to the Tutorials Chair, David Cook <david.cook@hill.af.mil>. Please submit questions on the conference to the Conference Chair, Salih Yurttas <yurttas@cs.tamu.edu>.

Deadline for Tutorial submissions: 6 May 2002; Deadline for other submissions: 3 June 2002

See SIGAda 2002 Home Page for details: http://www.acm.org/sigada/conf/sigada2002

Call for APIs

Pascal Leroy (Chair, Ada Rapporteur Group)
Rational Software Corp.

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

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

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

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

- *Benefits of the standardization:*
  Presumably the advantage of standardization is that it brings uniformity and portability among implementations. However, there is a significant overhead associated with a formal standardization process, so in some cases a *de facto* standard may bring practically the same benefits at a much lower cost.
- *Usefulness of the API:*
  APIs which have been conjured up solely for the purpose of writing a proposal, or which have been used by a very small group of users, are less likely to be generally useful than APIs which have been available for years and have benefited from feedback from a large user base.
- *Quality and precision of the proposal:*
  At a minimum, the proposal must include a set of Ada specifications, and a semi-formal description of the semantics of each declaration, such as can be found in the annexes of the Reference Manual. A rationale showing examples of use, explaining the choices that were made, the alternatives that were considered, and why they were discarded, would also be much appreciated.
- *Community consensus for the proposal:*
  Proposals with a substantial consensus of the Ada community or the appropriate subcommunity are preferred over proposals made by an individual or small group. This is not to say that a proposal primarily authored by an individual is necessarily bad (indeed, it is likely to provide a more consistent proposal), but to encourage authors to seek input/approval from as many potential users of the API as possible.
- *Portability and language usage:*
  The definition of the API must not depend on implementation-defined characteristics of a particular compiler, although it is acceptable to require the compiler to support some Specialized Needs Annex (or part thereof). As much as possible, the API should only use the features of Ada 95 (as opposed to those that are under consideration for the 200Y amendment) although we realize that this may not be practical in some cases.
- *Implementation:*
  A publicly available reference implementation would be useful, although this is not a strict requirement, as in some cases that may cause intellectual property issues.

Test suite. A test suite ensuring conformity to the specification should be provided at some point during the standardization process. This is especially important for standards for which no publicly available reference implementation will be available. This doesn't necessarily mean that there will be a formal conformity assessment process like there is for compilers, but it will help implementers ensure that they comply with the standard.

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

Some Impressions from IRTAW 11

John Barnes

11 Albert Road, Caversham, Reading RG4 7AN, UK.; Tel:+44 118 947 4125

Abstract

This paper gives a brief personal overview of some of the deliberations and happenings at the 11th International Real-Time Workshop held at Mont Tremblant in Canada in April 2002. Observe that this is a personal account; for the official version please consult the references.

Keywords: Ada, real-time, safety-critical.

Background

The real-time workshops were started by Ada UK at the suggestion of Mark Dowson who was then at Imperial Software Technology and later moved to the Software Productivity Consortium at which the Ada Quality and Style Guides were developed. I seem to recall that Mark having suggested the idea, I got lumbered with the job of running the first one which was held in May 1987 at the remote hamlet of Moretonhampstead on the edge of Dartmoor.

The main theme of that workshop was to obtain feedback from early experiences of the real-time features of Ada 83 with the twin aims of identifying workarounds and alterations. In other words, firstly suggesting how the existing features could best be used and, secondly, suggesting how the language might be improved in any revision.

The worries identified at that workshop included important matters such as priority inversion, asynchronous transfer of control, the initialization of task data and the need for a low-level mechanism to complement the high-level rendezvous. In due course these matters were fed into the discussions which resulted in Ada 95.

That workshop must have been successful because a second one was held at the same location the following year.

Incidentally, the structure established by those early workshops has remained. Each day is divided into two sessions with a long midday break so that the second session works on into the evening. This has a number of advantages. Attendees can take a long walk or do other physical exercise in the afternoon – the theory being that they can revitalise their mental powers by taking exercise and perhaps ponder matters with colleagues as they do so. It also means that by keeping on right up to dinner, there is no risk of frittering away the time in the bar before dinner. It has also been a tradition to hold the workshops in remote places so that attendees are not tempted to skive off to local museums or places of unhealthy entertainment.

Internationalization

Those early workshops in England had international audiences but the international nature of the workshops was subsequently strengthened by the third one being in the US and the fourth in Scotland.

The fourth workshop which was held at Pitlochry in July 1990 in fact provided a forum for the first general debate on what should be in Ada 95 concerning both tasking in particular and overall abstraction facilities in general.

The sixth and eighth workshops were held at another remote hamlet, namely, Ravenscar in Yorkshire; the dates were September 1992 and April 1997 respectively. An important outcome of the second of these was the Ravenscar profile which identifies facilities suitable for safety-critical and other high integrity systems.

I have omitted to mention the other workshops which I believe were held in the US and Spain because I was unable to attend them for various reasons. However, the purpose in mention the series as a whole is to emphasize that the workshops have been and continue to be an important focal point for discussing the evolution of Ada.

And so the eleventh workshop continued the tradition of being at a remote location, Mont Tremblant in Quebec. And it continued the tradition of seeking improvements to the Ada language with the prospect of a further revision in around 2005.

Location, location

Before addressing some of the technical issues, I feel obliged to make a few comments about the location which matters so much in making an event successful.

It was my first significant trip to Canada. I left England in glorious spring sunshine with the birds singing and the spring flowers springing up. I arrived in Montreal after wretched airline food to find grey skies, buckets of rain and a general dismal look about the place. After some time in a small bus looking at the continuing rain and lumps of snow and some grinding uphill we arrived at Le Club Tremblant.

Things immediately got better. The accommodation was excellent – and I also found it invigorating because it was 100++ steps up from the meeting and dining area to my room. Things got even better when we tackled dinner. The food was wonderful avec strong French influence. I think I would go so far as to say that overall it was some of the best food I have ever had in North America. We even identified some excellent Canadian wine after a few false starts. The weather got better as well.

Amendment to Ada 95

My main interest in the workshops is the identification of required changes to the language. I have observed that the workshops can be in different moods; sometimes looking at the current language in a reflective mood, sometimes looking to the next revision in an expectant mood. In view of the fact that the ARG is now in the process of identifying potential amendments for the next revision, it was appropriate for the workshop to be in an expectant mood. Accordingly, the workshop started with a presentation by Jim Moore, the convenor of WG9, on the international standardization process. He covered the structure of the various standards organizations and the way in which a standard is progressed with particular emphasis on the development of an Amendment rather than a Revision.

The amendment process is driven by the mechanism of Ada Issues (AIs). These are a well established route for dealing with corrections to the language (as evidenced by the 2000 Corrigendum incorporated into the recently published Consolidated Ada Reference Manual [1]). The development of AIs is hence also being used as the means for the production of the planned Amendment.

Existing Amendment Issues

The first issue to be discussed was that formalizing the Ravenscar profile (AI-249); this was essentially done at recent ARG meetings and the discussion was really just to tidy a few loose ends.

A related issue was the identification of why a task had terminated. It is a well known Ada curiosity that a task can suffer a silent death because of an unhandled exception. Ravenscar is silent on sequential aspects of the language in general and exceptions in particular and although they cannot arise for tasking reasons using the Ravenscar profile, nevertheless they might arise from the malfunction of other aspects of a system. In any event the problem affects programs in general and not just those confined to the Ravenscar profile. An AI addressing this issue through so-called task groups had already been devised (AI-266). This gives the programmer the ability to identify why a task has terminated and to perform last wishes. The general idea is that procedures are associated with various reasons for termination such as unhandled exception, normal termination, aborted and so on; these procedures are then called automatically by the runtime system when a task is terminated. Moreover, different procedures can be associated with different tasks or groups of tasks.

However, the overall view of the workshop was that although the objective of the existing AI was fine, nevertheless the style of the AI was somewhat too heavy and provided unnecessary flexibility (it was partly inspired by Java thread groups and it was reported that these had not been an uproarious success). The workshop accordingly devised a simpler approach which is being submitted to the ARG for consideration as an alternative.

Another AI which had originated from Ravenscar was AI-265 on partition elaboration. The problem here is that there are certification concerns regarding race conditions during the elaboration of library-level tasks. The AI proposes a partition elaboration policy which can be set by a pragma to either Sequential or Concurrent with Concurrent being the default and the existing behaviour. The workshop had concern with the applicability of the AI to task elaboration not at library level and it was agreed that some simplification to the AI would be appropriate.

Some time ago, an AI had been drafted on the concept of extended protected types (AI-250). The background is that Ada permits the extension of tagged record types but does not permit the extension of protected types. Superficially this seems attractive with the idea of being able to add further protected operations and so on. However, there are difficulties in the details. It was the overall view of the workshop that this was really not worth pursuing since no user need had been identified.

The final existing AI discussed by the workshop concerned exceptions as types (AI-264). The workshop revealed very mixed views on this topic. Some members felt that the existing proposal was an uncomfortable midway position, neither fully object-oriented nor basic. This reflected a range of views of user needs. Some applications forbid exceptions completely; others permit top level handlers only; yet others forbid propagation except on a "handle and reraise" basis and so on. It was concluded that although the workshop was sympathetic to the general objectives of the AI, nevertheless the matter was not of urgent concern.

Scheduling Ada Issues

A number of position papers prepared for the workshop were around the general theme of providing further scheduling capabilities. Perhaps the best way to give a flavour of the discussion is to outline some of the proposals which it was thought merited being formulated as draft AIs for consideration by the ARG.

There are situations where it is desirable to be able to dynamically change the ceiling priority of a protected object. However, such a change clearly has to be a protected operation. The proposed mechanism is to introduce a procedural attribute 'Set_Ceiling which takes the new ceiling priority as a parameter and can only be called in a protected procedure or protected entry of the object. Thus we might envisage

```
protected type Some_Type is
  ...
  procedure Change_Ceiling(P: Priority) is
  begin
    Some_Type'Set_Ceiling(P);
  end;
end Some_Type;
```

Note that this would introduce another situation where the type name (Some_Type in this case) is used inside the declaration of the type itself. There is a general rule that in such situations the type name refers to the current object.

Other examples like this include referring to the task type name inside the declaration of a task type – again in such circumstances the task type name refers to the current object.

Another interesting proposal is to introduce timing events. These would permit user-defined (protected) procedures to be executed at specified times without the introduction of tasks or delay statements. In the existing language a protected procedure can be associated with an interrupt event so being able to associate a protected procedure with a timing event would be a natural extension.

The approach suggested is to introduce a child package of Ada.Real_Time with specification

```
package Ada.Real_Time.Timing_Events is

  type Timing_Event is limited private;

  type Parameterless_Handler is
                 access protected procedure;

  procedure Set_Handler(TE: in out Timing_Event;
                At_Time: Time;
                Handler: Parameterless_Handler);

  ...
end Ada.Real_Time.Timing_Events;
```

Another subprogram Set_Handler enables events to be set for execution after a relative time rather than at an absolute time. Note that times naturally use the real-time clock since the package is a child of Ada.Real_Time. Other subprograms enable the state of an event to be interrogated and an event to be cancelled.

When the relevant time occurs, the protected procedure passed as parameter is executed.

For some safety-critical applications very simple cyclic executives are used and non-preemptive dispatching is required. Accordingly, it is proposed that (optional) new policies be introduced for task dispatching and locking so that we can use the existing pragmas thus

```
pragma Task_Dispatching_Policy(
      Non_Preemptive_FIFO_Within_Priorities);

pragma Locking_Policy(Non_Preemptive_Locking);
```

These policies have to be used together and then give the required non-preemptive dispatching.

Another scheduling topic which was discussed was CPU budgeting. This had been excluded from Ada 95 (although it was in the requirements) partly because of cost and partly because operating systems did not provide the required underlying support. However, using experience from POSIX, it was felt that the subject should be revisited and an AI would be drafted in the near future.

Other topics

The workshop covered many other general topics as well as the specific Ada Issues outlined above. These included fault-tolerance and distribution, experience with VXWorks, using partitions for security, software-related accidents, the revision of ARINC 653 and so on. However, in this brief summary I have concentrated on the Ada Issues because of their concrete nature and relevance to the planned amendment.

For fuller details of all aspects of the workshop please consult the Workshop Proceedings [2] which are being published as a special issue of Ada Letters.

Future activity

The workshop concluded by drawing up a timed action plan for revising existing AIs and generating new ones. Those on Ravenscar and termination were the most urgent and would be ready for the Vienna meeting of the ARG with the remainder being fully drafted by September.

The next workshop will be held at a location near Porto, Portugal in the week 15-19 September 2003. For details please contact the Program Chair, Tullio Vardanega; email tullio.vardanega@math.unipd.it.

Acknowledgements

I would like to take this opportunity to thank Stephen Michell for all his efforts in organizing the workshop in such an excellent location and with such good facilities.

Finally, I must thank Ada Language UK Ltd for their support without which I would not have been able to attend the workshop.

References

[1] S. Tucker Taft, Robert A. Duff, Randall L. Brukardt and Erhard Ploedereder (eds) (2000), Consolidated Ada Reference Manual; LNCS 2219, Springer-Verlag.

[2] Proceedings of the 11th International Real-Time Ada Workshop, Ada Letters (2002) (to be published).

Using Ada's Syntax and Semantics for Understandable Systems Engineering

Ingmar Ögren

Tofs corporation, Fridhem 2, SE 76040 Veddoe, Sweden Tel +46 176 54580; email: iog@toolforsystems.com

Abstract

Systems engineering is where you work with complete systems where operators cooperate with software and hardware modules to complete missions. This is of increased interest to software engineers in order to decrease the risk that the wrong software with the wrong requirements is built.
The present trend is to extend the Unified Modelling Language (UML) to cover not only software but also systems. This may lead to a risk for problems with understanding of system models because of many, partly overlapping, diagrams being used.
A proven alterative, which decreases this risk, is to use a few UML diagrams together with Ada 95 syntax, modified for use in Systems Engineering.

Keywords: Systems engineering, UML, Ada 95 .

1 Introduction

Systems engineering is the art of building systems where people cooperate with software and hardware to complete one or several missions.

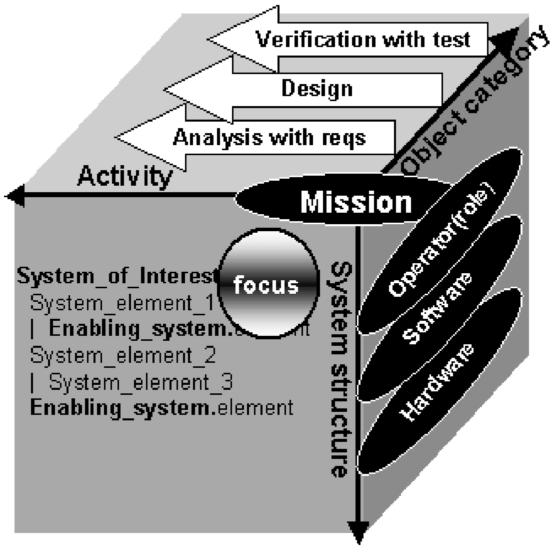

Figure 1: The Systems Engineering work space

Engineering of small systems is normally not a problem. Can often be done "on the side" as part of a software engineering effort. When systems are getting larger and the information grows beyond what is humanly manageable the situation gets different and the development group feels like moving in a "dark three-dimensional space", where you can only focus on a small part of the system at one time.

The three dimensions are (Figure 1):

- Activity, which defines what you do
- Object category, which defines what you are working with
- System structure, which defines where in the system structure you are working.

You may wonder what this has to do with software engineering and Ada. In fact a lot and the two disciplines depend on each other. Software engineering depends on correctly completed systems engineering, prior to programming, in order to ensure that you do not only produce correct software, but that you also produce the right software. This is extremely important since so much software is produced to great cost and then never used.

On the other hand software engineering has been around for a long time and systems engineering should benefit from using much of the hard-won experience from software engineering.

2 Requirements on a modelling language for systems engineering

A key activity in systems engineering is modelling, where you build a model of an existing or planned system. The more or less completed model can then be used for guidance of further work, including software development. Modelling is done in one or several modelling languages. Some of the key requirements on a modelling language are:

Understandability
: The is the most important requirement since systems engineering often involves several stakeholders with each of these being an expert in her own field and with little time to learn a modelling language. If the stakeholders do not understand and review the system model, someone will inevitably tell you after you believed the system was completed: "this is no good".

Structure
: Just like software, systems can be seen as a collection of co-operating modules with dependencies between the different modules. It is important that the modelling language can define the dependencies and show how the

different modules contribute to completion of the system's mission(s).

Behaviour
: Each module in a system, be it an operator role, a software or a hardware module, has a behaviour space, which defines its possible behaviour. Consequently system understanding requires the modelling language to be able to describe behaviour.

Communication
: Communication and interfaces is a key issue, not only for software, but also for systems work. Consequently a systems modelling language must model both "invocation style" communication and "message passing style" communication (between concurrent processes).

3 Today's solutions

Today various diagrammatic and textual languages are used for system modelling purposes such as IDEF0, Structured Analysis with Data Flow Diagrams, UML (Unified Modelling Language[2, 3, 4, 8]), Entity-Relationship diagrams and natural language.

These solutions all have their advantages, but also problems with the main problem being understandability. What is not always understood is the difference between teaching some students a modelling language for use on a small system and modelling a large and complex system and have the stakeholders concerned understand it although they have no time to learn the language used.

Today's trend is towards extended use of the UML, not only for software modelling, but also for modelling of systems. Several problems, with this extended use of the UML, have been observed, one of the most important being the language's difficulties to visualize "deep dependency structures". The solution to the problems, which seems to be most popular, is to introduce extensions to the UML. If you consider the understandability problems already present with the UML this may not be a possible road towards a useful system modelling language.

4 Ada 95 developed into a modelling language

When Ada was introduced in Swedish defence it was also used as a vehicle for introduction of software engineering. An Ada-based pseudo language the "Adel" (Ada Design Language) was then developed. When later the need for organized systems engineering was identified, Adel was further developed into "Odel" (Object Design Language[9]). Odel is still Ada-based, but compared to Adel it is more formal and allows for objects, other than software.

The Odel syntax is alphanumeric just like Ada 95 but it is closely connected to two UML diagrams, the Component diagram[1, 2] and the Message Sequence diagram [2].

The result is a modelling language which meets the requirements listed above as follows:

Understandability
: Understandability is ensured through a combination of simplified Ada95 syntax (Formalized English) and simplified UML Component diagrams (Object graphs), drawn from the Odel descriptions.

Structure
: The structure in the Odel descriptions is based on Ada's "withing principles" developed into dependency structures, including object (types) of categories Mission, Operator, Software and Hardware. After an idea from Håkan Lindegren of Örebro University the "deep dependency structure" is visualized in "Tree graphs".

Behaviour
: Behaviour is modelled in a simplified and extended Ada 95 syntax, with the main modifications being:

  - Procedures, Functions and Tasks are combined into "Actions" after an idea from professor Vitalis S Kaufman of Moscow State and Tammerfors Universities.
  - Concurrency between concurrently active actions is described through a "concur" statement after an idea from Ingalill Bratteby-Ribbing of the Swedish Defense Material Administration.

Communication
: Communication in system modelling is done basically in two ways: through invocation, with parameter passing, and through message passing between concurrent processes. Invocation is no problem with use of Ada's principles for parameter passing. Message passing is different, but a solution was found, with introduction of "messages", based on Ada's principles for management of global variables and with introduction of the reserved words **send** and **receive**.

The Odel language was defined in the style of the Ada 95 Language Reference Manual and primarily through references to the Ada 95 LRM. The Odel Language Definition is freely available to anyone interested. To request it, send a mail to info@toolforsystems.com

3 An example

As an example consider a system called "Car window control" (Figure 2) with the mission to control the windows in an automobile and, besides the mission object, containing objects of categories operator (Driver), Software (Central window control and Local window control) and Hardware (buttons and switches). Note further in the Tree graph:

- Each object has a "little clock", which indicates its development status
- A system is defined as a "Configuration Item" (CI), in accordance with ISO/IEC 12207 standard[ref 6]. The different system-related concepts in the ISO/IEC 15288 standard are all managed as CIs. In figure 2, besides the current CI "Car window control", you can

see an attached CI "Door", representing an "enabling system in the wording of the ISO/IEC 15288[7].

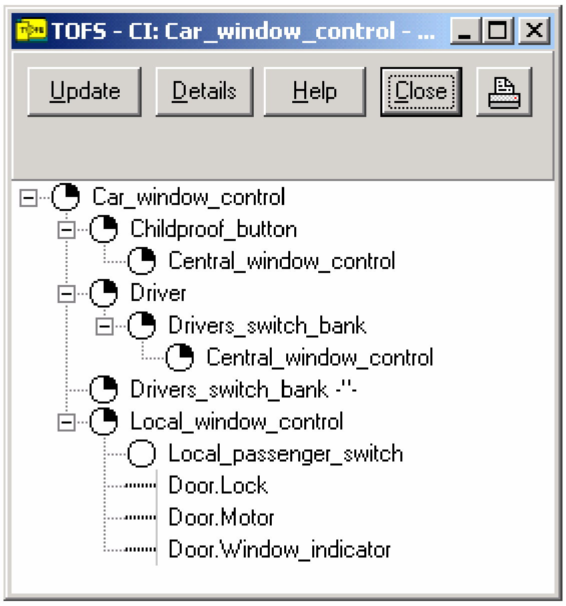

Figure 2: Tree graph for "Car window control"

If you focus on the mission object "Car window control", it can be expanded into a component diagram (Figure 3). For simplicity reasons arrows are omitted and only two levels are shown.

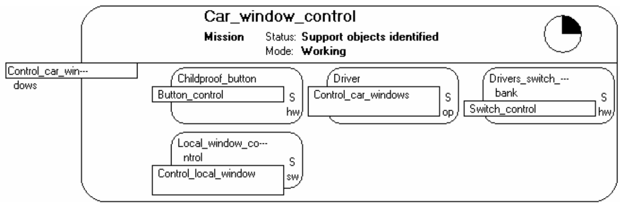

Figure 3: Top component diagram (object graph) for "Car window control"

Note that this diagram also shows the offered and required interface for the Car window control object:

- In the left hand "action box" is shown that the object offers the action "Control car windows"
- Each support object shows action(s) in their "action boxes". Together these constitute the required interface for "Car window control".

If you go into the action description for the action "Control Car windows" you find a "concur" statement, which defines that the actions in the support objects shall be active concurrently:

```
concur
 # Driver.Control_windows
 # Drivers_switch_bank.Switch_control
 # Childproof_button.Button_control
 # Local_window_control.Control_local_window
end concur
```

If you look at the component diagram for the "Local window control" object (Figure 4), you can see how objects from attached CIs are drawn "outside" in the tradition from HOOD (Hierarchical Object Oriented Design) syntax [5].

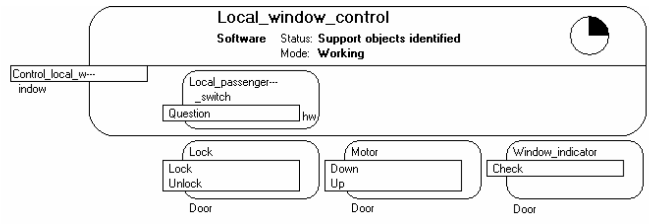

Figure 4 Component diagram with objects from an attached CI

In the example the two "window control" software objects are active concurrently and communicate by way of messages. The Odel fragment below shows how an action for sending such messages can be described. In the other software object a corresponding action will be found with reception of the "up" and "down" messages.

Note that the Odel description is formal enough to be analysable for correct syntax and to be useful to guide a programmer. It may not be directly understood by all stakeholders, but should be understood, after a short introduction, because of its closeness to natural language.

```
action Window_command
 (direction_up : in Boolean,
 window_concerned : in seat_position) is

visibility: Offered

purpose: { Receive orders to control windows and
forward these by way of the CAN bus }

messages:

up_message,  down_message

variables:

begin
 case direction_up is
   when true  => send up_message(window_concerned)
   {send message to move window upwards to the local
   control software for the concerned window in the car,
   defined by seat position
   (driver, other_front, rear_left or rear_right)}
   when false  =>
     send down_message(window_concerned)
   when others => null
 end case
end
```

3 Conclusions

The principles described above have been applied in several industrial and defence project with the experience that the combination of a few diagram types and Ada 95 based textual descriptions gives a system modelling language, which is both understandable to stakeholders concerned and useful as a formal basis for design of software and hardware modules.

The main advantage, when compared with a completely diagrammatic alternative, such as "extended UML" is the greater simplicity and the resulting better understandability. This is important, since better understanding by stakeholders should decrease the risk of building the wrong software (which is never used).

Another advantage is that some formality is introduced, through the coupling to Ada 95, without introduction of "mathematical syntax", which may introduce new problems to understand the descriptions.

References

[1] Booch, Grady, Software Engineering with Ada, Benjamin Cummings 1983

[2] Fowler, Martin, UML Distilled, Addison Wesley 1999

[3] Ogren Ingmar, On Principles for Model-based Systems Engineering, *Systems Engineering*, Vol 3, No 1, pp. 38-49, 2000

[4] Ogren, Ingmar, Possible Tailoring of the UML for Systems Engineering Purposes. *Systems Engineering*, Vol 3, No4, pp. 212-224, 2000

[5] Rosen, Jean-Pierre, HOOD An industrial approach for Software Design, HOOD technical group 1997

[6] www.12207.com, the website that gives information on the standard for Information Technology –Software Life Cycle processes

[7] www.15288.com, the website that gives information on the standard for Systems Engineering –System Life Cycle processes

[8] www.omg.org, the web site where you find UML-related document, including definition of the "Component diagram"

[9] www.toolforsystems.com, the website which contains a downloadable version of the present release of the Tofs system modeling software, including an analyzer for the Odel language

Real-time Programming Safety in Java and Ada

Bo I. Sandén

Computer Science, Colorado Technical University, 4435 N. Chestnut St. , Colorado Springs, CO 80907-3896, U.S.A. Email: bsanden@acm.org   Phone: (719) 590-6733   http://iis-web.coloradotech.edu/bsanden

1. Introduction

Few industry-strength languages include multi-threading in their syntax. Among contemporary languages, Ada [1, 2] and Java [3-8] are the most prominent. The philosophy of concurrency is similar in Java and Ada95 and is based on the classic distinction between threads (tasks in Ada) on the one hand, and shared objects on the other. This has been the dominant paradigm for practical multi-threading for decades, although other models exist, such as the rendezvous paradigm of Ada83, which is still supported in Ada95. (The Ada83 paradigm will not be further discussed here.) In Ada, shared objects are declared "*protected*". In Java, they are instances of classes that have methods marked "*synchronized*".

Although the philosophy of concurrency is similar, the attitude to safety and reliability is radically different in Ada and Java. Java threading is adequate for its original purpose: windows programming and applets. But Java is now being fitted with real-time extensions and may be applied to safety critical software. The language has many potentially abusable constructs and programmer pitfalls. This is certainly true for Java in general [9] but is particularly important with concurrent software, which is notoriously difficult to debug.

The Real-Time Specification for Java (RTSJ [10, 11]) does nothing to remove the pitfalls. It intends to make Java useful for real-time applications by circumventing the garbage collector and providing interrupt handling, not to make the language less error prone. The case that Ada95 is a much safer language for real-time embedded applications than Java can easily be made.

This paper is intended for Ada programmers, who may be taking the Ada concurrency features for granted. The purpose is to view those features against a backdrop of the pitfalls of a more traditional concurrency implementation without emphasis on safety. I point out a number of Java threading pitfalls and note how they are prevented in Ada. While I briefly summarize the Ada tasking model, the reader is assumed to have an understanding of Ada tasking and sufficient understanding of Java to be able to read small program excerpts, but is not expected to know much about Java multi-threading. For a comprehensive comparison of concurrency features in the two languages, see [12].

1.1. Forms of synchronization

The traditional concurrency model with threads on the one hand and shared objects on the other relies on a distinction between *exclusion synchronization* and *condition synchronization*, described as follows.

Exclusion synchronization is used to stop two threads from operating on the same object at the same time and thereby jeopardizing the integrity of its data. Java provides exclusion synchronization for any synchronized method. Ada provides exclusion synchronization for protected operations. A block of code that is executed under exclusion synchronization is called a *critical section*. It is bracketed by instructions that acquire and release a lock on an object.

Each thread is expected to maintain exclusive access for a very short time, making it unlikely that a thread will ever find an object locked. If it does find an object locked, only a brief wait should be expected. It is even more unlikely that two threads should attempt access to the same object while its is locked. For this reason, one need not be concerned with maintaining a orderly queue of threads pending on an object lock. Implementations of exclusion synchronization typically let a thread that encounters a locked object yield the processor in the hope that it will find the object unlocked when next made running. If the object is still locked, the thread again yields the processor. With multiple processors, one often uses a *spin lock*, that is, the thread enters a loop where it repeatedly attempts access until it is successful. I shall use the term "spin lock" for both the single processor and multi-processor cases.

While a thread, *l*, is operating on a shared object *O*, under exclusion synchronization, it may be preempted by a higher-priority thread, *h*, which also needs exclusive access to *O*. Unavoidably, *h* must wait for *l* to exit the critical section. Such a situation where a higher priority thread is waiting for a lower priority thread is referred to as *priority inversion*. If *l* continues executing at its normal priority, a thread, *i*, of intermediate priority may preempt *l*. This leads to an avoidable situation where *h* is waiting for two lower-priority threads. To avoid it, *l* can be given a priority boost. One possibility is to let *l inherit h*'s priority once *h* tries to access *O*. The other possibility is to define a *ceiling priority* for *O*, which is used by any thread while it executes a synchronized method on *O.* The ceiling priority must be as high as the priority of any thread that ever operates on *O*.

Condition synchronization is when a thread cannot proceed if a certain condition holds. A buffer, shared by two or more threads provides a classic example. A *Buffer* object has the operations *put( ),* called by producer threads, and *get( ),* called by consumer threads. A thread that calls *put( )* must wait if the buffer is full, and a thread calling *get( )* must wait if it is empty. There is no assumption that this wait will be brief. Threads may spend considerable time waiting, and must be queued, typically first-in-first-out per priority. A thread conditionally waiting for an object never holds the object locked and should not normally hold other objects locked.

Condition synchronization must be used to control access to any shared resource that is held long enough for a queue of waiting threads to form. Examples of resources of this nature range from a printer or a database record to resources in the problem domain of a control system such as railroad segments and automated vehicles in a flexible manufacturing system. Access to such a resource is handled by means of an object with a Boolean variable *busy*, say, and operations such as *acquire( )* and *release( ).* Once a thread has successfully acquired the resource and set *busy* to true, it releases the object lock, allowing other threads to call *acquire( )* and place themselves on queue.

The term "condition synchronization" gives no hint that condition synchronization is often used to control access to domain resources and, in general, to resources held for a long time. The terms "competition synchronization" and "cooperation synchronization" for exclusion and condition synchronization respectively are no better [13]. The rationale here is that the producer and consumer threads must cooperate to manage an empty and a full buffer but compete over the access to the operations. Again, this ignores the use of condition synchronization to control exclusive access to a domain resource. An alternative way to characterize synchronization is to distinguish between exclusive access with *short extent* in time (exclusion synchronization) and *long extent* (condition synchronization) [14].

The distinction between exclusion and condition synchronization is crucial in real-time concurrent programming. While Ada and Java both support exclusion and condition synchronization, Ada helps the novice programmer by clearly separating the concepts syntactically. Java does not, and some of the pitfalls result from a confusion of them.

2. Ada concurrency model

Ada95 implements concurrency with two kinds of entities: *tasks* and *protected objects*. You define a task type, which is instantiated as any other type, or a singleton task. *Protected objects* have monitor-like behavior. They can have *protected operations* of three kinds: *functions*, *procedures* and *entries*. These are declared in the specification of the protected type[1] as for example the following:

```
protected type X is
  function F1(  ) return Type1;

  procedure P1 (  );

  entry E1 (  );

  private

    - -  Attribute variables

    - -  Private operations including interrupt handlers

end X;
```

All protected operations have exclusion synchronization built in. The differences between protected functions, protected procedures and entries are as follows:

Protected functions are read-only. They are prohibited from changing the attribute values of the protected object and are subject to a read lock: Any number of function calls on a given object are allowed simultaneously, but not during a procedure or entry call on the object.

Protected procedures are allowed to change attribute values. They are subject to a write lock: Only one procedure (or entry) call at a time is allowed on a given object, and not during any function call.

Like procedures, entries are allowed to change attribute variable values and are subject to the write lock. In addition, an entry can provide condition synchronization by means of a barrier condition, which appears in the body of the protected type. An entry call only proceeds when the condition is true. For example, an entry Get, which is only allowed when the number (Num) of items in a buffer is greater than zero, may be declared as follows:

```
entry Get ( ...  ) when Num > 0 is ....
```

A task that calls Get when Num = 0 is put on a queue that is FIFO per priority. Each protected object has one queue per entry.

Any variables in the barrier condition are supposed to be attribute variables of the protected object, on which the entry operates. The values of those variables can only be changed by calls to protected procedures and entries on that object.

[1] Ada also provides for singleton protected units.

At the end of each procedure or entry call on a given object, its barriers are evaluated. If a barrier is found to be true, the most eligible task in the corresponding queue is activated and executes the entry body. Tasks that are already in a queue have precedence over new callers according to the principle of "internal progress first".

3. Java concurrency model

In Java, any method in any class can be declared *synchronized*. This is exclusion synchronization: A write lock per object is applied, so that only one synchronized method at a time can operate on a given object. A synchronized method in Java is similar to a protected procedure or entry in Ada in that it is implicitly bracketed by instructions that acquire and release the object lock.

Condition synchronization in Java relies on explicit tests programmed into the synchronized methods, as for example:

```
while (0 == Num) {wait( );}
```

If Num is zero, a calling thread calls *wait( )* and thereby enters the object's *wait set*. There is one wait set per object, not one per entry per object as in Ada. A thread, *t*, that executes a synchronized method on an object may change the truth value of a condition that may affect one or more threads in the wait set. Before leaving the method, *t* must explicitly *notify* any such threads. The call *notifyAll( )* reactivates all threads waiting for conditional access to an object.

The Java thread model hides little from the programmer. This makes it quite flexible for the old hand at concurrency. Apart from the tie-in with object-orientation, the thread model is in fact very similar to what I personally encountered when manipulating threads provided by the UNIVAC 494 operating system in assembler programs 30 years ago. In a sense, this makes Java more pedagogical than Ada because it exposes the details of synchronization. But by the same token, the Java model is much more error prone. Very little protects Java programmers from the consequences of their own mistakes. Unfortunately, many programmers are not shy about trying things they don't fully understand and then testing the program to see if it works. Many concurrency related bugs are subtle enough to pass most tests. An Ada programmer cannot easily make clerical errors with unintended effects on the program behavior.

An advantage of the Java model is that it can be implemented with less overhead than the Ada model, but this issue appears to be losing much of its earlier importance. Further, a Java class with synchronized operations can be part of the inheritance hierarchy. In Ada, this cannot be done directly. Although Ada 95 includes object-oriented features including inheritance, polymorphism and dynamic binding, these were not extended to protected objects [15].

3.1 Real-time Java

The Real-Time Specification for Java (RTSJ) [10, 11] is an effort to make Java useful for real-time programming. (An alternative specification is given in [16].) One premise is that a real-time program must be predictable so that the programmer can determine *a priori* when certain events will occur. This is not true for standard Java for a number of reasons. First, the garbage collector, which can interrupt any other processing, adds an element of randomness. Second, different scheduling policies cannot be imposed in standard Java. (Scheduling policies such as the rate-monotonic algorithm allow you to prove that a set of threads meet their specified deadlines.) Third, in standard Java, threads placed in a wait set are reactivated in arbitrary order, independent of when they attempted access; the wait set is not a FIFO queue.

To deal with these problems, RTSJ introduces a number of new classes, most of which are necessary for working around the garbage collector. One such class is NoHeapRealtimeThread (NHRT), which is a descendant of Thread. NHRT threads have higher priority than the garbage collector so are not subject to arbitrary delays. This places many restrictions on the programmer, however. For example, an NHRT thread cannot allocate objects on the heap. Instead, RTSJ provides for various kinds of special memory areas.

RTSJ also stipulates that threads in a wait set must be kept in FIFO order within priorities. This means that notify( ) reactivates the thread with the highest priority. If there are more than one thread with that priority, the one that has waited the longest is reactivated.

RTSJ uses priority inheritance as the default control policy to address priority inversion. A priority ceiling protocol is also specified. Finally, to further support real-time programming, RTSJ allows the programmer to specify interrupt handlers.

4. Java pitfalls and their Ada solutions

Apart from the extensions provided by RTSJ, real-time Java relies on the threading and synchronization models of standard Java. Next, I discuss in more detail some features of these models from a real-time point of view, focusing on the associated pitfalls. I also discuss how each pitfall is prevented by the Ada95 syntax and semantics. Although programming to RTSJ is in many respects quite involved, I do not address any programming pitfalls that may appear there.

4.1 Defining and starting threads

Java provides the abstract class *Thread*, whose method *run( )* represents the logic that a thread performs. It corresponds to the executable part of a task body. A standard way of creating threads is to declare a new class, *T*, that extends *Thread* and overrides *run( )* with appropriate processing. Each instance *To* of *T* has its own thread, which is explicitly started by means of the call *To.start( )*. Once started, the thread executes *T*'s *run( )* method and has access to *To*'s data.

Because Java has no multiple inheritance, an additional mechanism is necessary for the case where a class, *R*, that needs a thread, already extends another class, such as *Applet*. For this situation, Java provides the interface *Runnable*. The programmer makes *R* extend *Applet* and implement *Runnable*. Instantiating *R* creates a *runnable object*, *Ro*, say. To associate a thread with *Ro*, you submit *Ro* as an argument to one of *Thread*'s constructors and then call *start( )* on the resulting *Thread* instance. This is typically done in a statement such as:

```
new Thread(Ro).start( );
```

Java pitfalls. Once you have a class *R* that implements *Runnable*, Java gives you two ways to create multiple threads that execute *R*'s *run( )* method. First, you can instantiate *R n* times, and submit each instance once as a parameter to one of *Thread*'s constructors. Now you have *n* instances of *R*, each with a thread, so each thread has its own set of instance variables. But Java also lets you submit the same instance of *R* repeatedly to *Thread*'s constructor. The result is a subtly different case where multiple threads are tied to one object and share its instance variables. Two (or more) of these threads can directly manipulate those instance variables simultaneously, and possibly introduce inconsistencies.

Ada. Ada's model for defining and starting tasks is cleaner. You declare a task type, which is instantiated as any other type, or a singleton task. The task is started automatically, either when the first executable statement is reached after the declarations, or, if a task type is dynamically instantiated, immediately upon instantiation. Each task instance has its own private data.

4.2 Synchronized objects

Java provides all objects with the potential for monitor-like behavior, that is, approximately the behavior of a protected object in Ada. Every object has a lock variable, which is hidden from the programmer and cannot be accessed from methods on the object. Exclusion synchronization is accomplished by specifying a method as *synchronized*, as in:

```
void synchronized m( ) ...
```

When a synchronized method is called on some object, *O*, its code is implicitly bracketed by statements that acquire and release the lock on *O*. That is, a thread calling *O.m( )* locks *O* as a whole, so that no other thread can perform any synchronized method on *O* (or execute any block synchronized with respect to *O* as discussed below). This synchronization feature is built in, which guarantees that the lock is always released when a thread leaves a synchronized method, even if this happens by means of the exception handling mechanism. I shall refer to any instance of a class that has at least one synchronized method or synchronized block as a *synchronized object*.

A Java programmer can choose to specify some but not all the methods of a class as synchronized. This has some useful applications. For example, a method that returns a constant or the value of a single attribute need not be synchronized.

Java pitfalls. The freedom to specify selected methods of a class as synchronized opens the door for mistakes. In the buffer example, the programmer may declare *get( )* synchronized and not *put( )*. This allows different threads to call *put( )* simultaneously. These calls may also overlap with a call to *get( )*. This jeopardizes the integrity of the buffer data structure. The program may still work much of the time, but will produce occasional errors, especially when run on a symmetric multi-processor providing true parallelism. Such errors tend to be hard to find by testing – although more easily by inspection by an experienced thread programmer. Omitting the keyword synchronized altogether, for *put( )* as well as *get( )* would further exacerbate the situation.

A programmer must ensure that the instance variables that the synchronized methods operate on are private so that they cannot be directly accessed and changed by a method operating on some other object. Even if they are private, you must ensure that they are not changed by a static method defined for the class.

Ada. All operations on a protected object require either a read lock or a write lock. You cannot include a non-protected operation in a protected object. A protected function can be used to return constants, etc., as can an unsynchronized method in an otherwise synchronized Java class.

4.2.1 Synchronized blocks

In addition to synchronized methods, Java provides synchronized *blocks*, which have no Ada counterpart. Any block in any method can be synchronized with respect to an object – not necessarily the current instance of the class where the method appears – by means of the syntax:

```
synchronized ( Expression ) {  /* Block B */  }
```

Expression must evaluate to a reference to some object, *Vo* of class *V*, say. As for synchronized methods, exclusion synchronization is implicit, so *B*'s code is bracketed by statements to acquire and release the lock on *Vo*. A synchronized method and a synchronized block are both critical sections.

Consider first the case where *B* is part of some method, *m( )*, on class *V* and is synchronized with respect to the current object as follows:

```
class V ...
{       void m( )
        {       synchronized (this)
                {       /* Block B*/
                }
        }
}
```

This design is an alternative to making *m( )* synchronized and can be used if only parts of *m( )* requires exclusive access. That way, two or more threads can simultaneously execute those parts of *m( )* that are outside *B*, so concurrency may be increased. An alternative design is to make *B* into a separate, synchronized method called from within *m( )*.

As mentioned, the block *B* in *m( )* can be synchronized with respect to any object, not only the current one. In the following excerpt, B is synchronized with respect to object *Wo* of class *W*. This means that before entering the block *B*, the thread that called *m( )* in order to operate on an object of class *V* acquires the lock on object *Wo* of *W*.

```
class V ...
{       void m( )
        {       synchronized (Wo)
                {       /* Block B*/
                }
        }
}
```

Synchronized blocks are useful when different threads need exclusive access to some object in order to perform their own, particular operations on it. Such an object is sometimes a printer or a window to which many threads write their own tailored outputs such as log entries as in the following example:

```
synchronized (myPrinter)
{
   // series of statements producing output
}
```

In this case, it can be inconvenient to make every possible combination of output statements into a method for the printer class.

Java pitfalls. By synchronizing blocks with respect to some object you effectively create “distributed” methods that are not included with other instance methods in the class definition. From looking at a class definition, you cannot tell whether any blocks exist that are synchronized with respect to its instances. A class without synchronized methods in its definition may appear to a maintenance programmer as an unsynchronized class.

Ada has no equivalent to synchronized blocks. All operations on a protected object are specified in its declaration.

4.3 Condition synchronization

The most common idiom for condition synchronization in Java is the statement

```
while (cond) {wait( );}
```

This statement makes the calling thread wait as long as *cond* holds. I shall refer to the statement as a *wait loop*. If *cond* is true, the thread calls *wait( )* and thereby places itself in the *wait set* of the current object, *O*, and releases *O*. The wait set contains all threads waiting for *conditional access* to *O*.

The wait loop syntax is somewhat complicated by the need to handle an *interrupted* exception that can be caught by a thread while it is in the wait set. This is possible because this particular exception is thrown by a different thread than the one that must catch it. Unless the exception is propagated to an enclosing scope, a construct such as the following is necessary:

```
while (cond) try {wait( );}
   catch (InterruptedException e)
   {   /* Take action or ignore the exception */
   }
```

Pitfalls. The wait loop is like an incantation that should always be repeated in almost exactly that form. For example, the variation

```
while (cond) {yield( );}
```

stops the calling thread from proceeding against *cond* but does not release the object. This means that other threads that are supposed to change *cond* by calling synchronized methods on the object cannot do so.

A more insidious mistake is to replace the wait loop with the quite similar statement

```
if (cond) {wait( );}
```

This statement makes the calling thread enter the wait set and release the object, but only once. When reactivated, the thread continues after the wait( ) call and proceeds in the synchronized method even if cond is true [12]. This is particularly dangerous, since notifyAll( ) must often be used and relies on the wait loop. When notifyAll( ) activates a thread that is not to proceed, the thread is supposed to retest its condition and return to the wait set. Substituting if for while leads to a typically transient error. Under unlucky circumstances, it can remain undetected for some time, perhaps until an additional thread is introduced.

Ada. Condition synchronization is achieved by means of entry barriers, which are built into the syntax. Their format is not susceptible to easy programming mistakes. One possible mistake is to include in a barrier condition a variable that is defined outside the protected object. In that situation, it is possible to change the value of the condition without notifying waiting threads.

4.3.1 Placement of the wait loop

The wait loop in Java most often appears at the very beginning of a critical section and is reached immediately after a thread locks the object. But it can be placed anywhere within a synchronized method or block. As a simple example of one or more statements separating the wait loop from the beginning of a method, you could count the number of calls to a method, *m( )*, in an instance variable *CallCounter* in the following way:

```
synchronized void m( )
{
   CallCounter ++;
   while (cond) {wait( );}
   . . . .
}
```

Here, *CallCounter* is incremented exactly once for each call, no matter if the calling thread enters the wait set. Its value equals the number or calls to *m( )* including those where the thread is still in the wait set. In a slightly more sophisticated example, the statements before the wait loop could maintain a list of the thread identities of the latest *n* callers. One can also instrument the wait loop itself similarly by including statements before and/or after the *wait( )* call.

The textbook case for placing the wait loop deeper inside a critical section is when a method allocates resources to calling threads. It may turn out that the request of a calling thread, *t*, cannot be satisfied until additional resources become available. In that situation, *t* can place itself in the wait set, release the object and wait to be notified by a thread that has released resources. Once notified, *t* continues immediately after the *wait( )* call with exclusion synchronization in force. If a synchronized method has one or more such *wait( )* calls, a thread can effectively execute it in segments separated by those calls, entering a new segment each time it is successfully reactivated from the wait set.

Java pitfalls. The syntactical freedom to place the wait loop anywhere in a critical section allows certain errors. Even if the wait loop is initially placed at the very beginning of the critical section, a maintainer can unintentionally insert statements between the beginning and the wait loop. These statements are executed exactly once by every thread that attempts access to the critical section regardless of the condition. This may be even more treacherous if there are already statements between the beginning of the critical section and the wait loop, as in the CallCounter example. The maintainer may not realize the difference in status between statements placed before and after the wait loop.

Ada. Because protected objects in Ada are syntactically distinct, there is no easy way to include a statement such as CallCounter := CallCounter + 1; in an entry in such a way that it would be executed exactly once under exclusion synchronization before the barrier has been passed. (Certain elaborate maneuvers are possible if you include in the barrier condition a function call with side effects.)

An Ada entry body cannot be broken into segments where a task would execute a segment, then release the object, put itself on queue and continue with the next segment upon reactivation. In Ada, each such segment must be an entry. A task that is executing an entry can call *requeue* and place itself on the queue of the same or another entry [2]. Requeuing is sometimes considered an advanced Ada topic. An intuitive example of requeuing when a resource turns out to be unavailable is given in [17].

4.3.2 Notification of waiting threads

A Java thread that executes a synchronized method on *O* and changes a condition that may affect one or more threads in *O*'s wait set must *notify* those threads. In standard Java, *O.notify( )* reactivates *one* arbitrarily chosen thread, *t*, in *O*'s wait set. If the call is correctly placed within a wait loop, this means that *t* reevaluates the condition and either proceeds in the synchronized method or reenters the wait set. In RTSJ, the most eligible thread is reactivated.

The call *O.notifyAll( )* releases *all* threads waiting for conditional access to *O*. This is useful when a condition has changed so that multiple threads can proceed. But calling *notifyAll( )* instead of *notify( )* is sometimes necessary even though you want only a single thread to proceed. In standard Java, this is the only way to give preference to the highest priority thread. It is inefficient if there are many threads in the wait set, since they must all attempt access, and only one will succeed [18].

Because there is only one wait set per object, you must also call *notifyAll( )* instead of *notify( )* if an object's wait set may include threads pending on different conditions. If you change one of the conditions, you must activate all the threads to make sure that a thread pending on that condition is notified, if it is in the set. This is true in RTSJ as well as in standard Java.

When a thread calls *wait( )*, *notify( )* or *notifyAll( )* on an object, it must have the object locked. The wait set is a shared data structure that must be protected from conflicting access, but has no lock of its own.

Java pitfalls. Unlike exclusion synchronization, condition synchronization is not automatic; you have to explicitly notify waiting threads. An obvious pitfall is to forget to insert *notify( )* calls at all the necessary places. This is particularly treacherous if a method has unusual exits, as via exception handlers. A related mistake is to call *notify( )* instead of *notifyAll( )* when threads in the same wait set may be pending on different conditions.

A way to reduce the risk of forgotten notifications is to include a timeout parameter in every *wait( )* call. After the given time, the thread is activated, and if the *wait( )* call is placed inside a correct wait loop, the thread reevaluates the condition and either proceeds or reenters the wait set.

Ada. As long as the entry barrier depends only on variables local to the protected object, the most eligible, waiting thread is automatically activated after a protected procedure or entry call on the object has changed the truth value of the condition. Waiting tasks are queued per entry, so precise notification can be achieved: If a single condition is changed, only a task waiting on that condition is activated.

4.3.3 Controlling access to domain resources

Condition synchronization is used to give one thread at a time exclusive access to a shared resource in the problem domain, such as a forklift truck in an automated factory application [14, 19, 20]. In this example, jobs on the factory floor that need the forklift are represented by *Job* threads in the software. A forklift operation may continue for several minutes and must be performed under condition synchronization because we want waiting jobs to form a FIFO queue per priority. The object controlling the forklift – instance *F* of class *Forklift*, say – typically has an attribute, *busy*, that reflects the availability of the forklift, and the synchronized operations *acquire( )* and *release( ),* where *acquire( )* contains a wait loop such as the following:

```
while (1 == busy) {wait( );}
```

The corresponding notification call is in *release( ).* Statement sequences where the forklift is operated are bracketed by calls to *acquire( )* and *release( )* whether they appear in *Job*'s *run( )* method or in other unsynchronized methods.

While one job is using the forklift, other *Job* threads can call *F.acquire( )* and place themselves in *F*'s wait set. The variable *busy* serves as the lock on the physical forklift while *F*'s hidden lock variable only serves to control the access to the variable *busy* itself.

Explicitly calling *acquire( )* and *release( )* in this fashion is similar to working with a semaphore, and may be counterintuitive if you have been taught that semaphores are a primitive way of controlling the access to a shared resource. A synchronized method or block is a more abstract representation that hides the semaphore operations. But when controlling access to shared resources in the problem domain in this fashion, we must invert the abstraction by using a synchronized object to implement a semaphore [20].

In the example with the shared printer, we can choose whether to consider the wait to be of long or short extent. In the solution in section 4.2.1, each thread's operations on the printer are enclosed in a synchronized block as follows:

```
synchronized (myPrinter)
{
  // series of statements producing output
}
```

This exclusion synchronization assumes that the printer operations are quick. If other threads try to access the printer during the exclusive access, they spin, waiting for the lock. There is no direct way to ensure that they will ultimately access the printer in a first-in-first-out fashion.

In an alternative solution based on condition synchronization, you define *acquire( )* and *release( )* methods in the *Printer* class (or another class), introduce a variable such as *busy*, and bracket the series of statements with calls to those methods:

```
myPrinter.acquire( );
  // series of statements producing output
myPrinter.release( );
```

Here, *acquire( )* contains a wait loop, and threads that must wait for the printer enter the wait set. A downside is that this solution makes the programmer responsible for inserting one or more *release( )* calls to ensure that the printer is released even if an exception is thrown while the output is being produced.

Java pitfalls. By convention, all critical sections should be programmed to minimize the time an object is held locked. A thread that is waiting on a condition should release its object locks and be placed in a wait set. But nothing stops a programmer from making a thread hold an object lock for an arbitrarily long time. A trivial way to do this is to call *sleep(...)* inside a synchronized method. Two other cases are described next.

Controlling domain resources. The confusion of long and short waits may typically occur in real-time applications that control resources in the problem domain. In the automated factory domain, the forklift operation may be implemented by means of the following synchronized block within the *run( )* method of the *Job* class:

```
synchronized (F)
{
  // Operate the forklift
}
```

This ensures mutual exclusion of jobs using the forklift and may at first seem more elegant than the solution with semaphores. But if the forklift operation continues for minutes, *Job* threads that need the forklift are not put in a wait set (and FIFO queued per priority in RTSJ) but spin until they find *F* unlocked. Which *Job* thread gets to the forklift next is then quite arbitrary. To avoid this, condition synchronization must be used.

In RTSJ, exclusion synchronization invokes the control policy to minimize the effect of priority inversion. Assume first that the default policy, priority inheritance, is in effect. If a job at priority *l* is currently operating the forklift and a higher priority job, *h*, attempts to get the lock, *l*'s remaining forklift operations will be executed at priority *h.* This skews *l*'s priority relative to any jobs with priorities between that of *l* and that of *h*. The ceiling priority protocol has an even more fundamental effect in that all forklift operations will always be carried out at the highest priority of any job.

Nested synchronized blocks. Another way of inadvertently mixing long and short waits is with nested critical sections. We can insert a wait loop in a nested synchronized block as follows:

```
synchronized(r1)
{
    ....
    synchronized(r2)
    {  while (cond) {r2.wait( );}
         ....
    }
}
```

If *cond* is true, the calling thread enters *r2*'s wait set and releases *r2*. But it keeps *r1* locked, and lets other threads that need access to *r1* spin rather than wait in a wait set. Incidentally, the following is also legal:

```
synchronized(r1)
{
    ....
    synchronized(r2)
    {     while (cond) {r1.wait( );}

            ....
    }
}
```

In this case, the calling thread enters r1's wait set and releases r1 while keeping r2 locked. On the other hand, placing a wait loop in the outer synchronized block is harmless as long as the block doesn't represent some lengthy operation such as the forklift operation discussed earlier.

Ada. The Ada syntax is certainly clearer about the distinction between exclusion and condition synchronization. Any protected operation provides exclusion synchronization automatically. Any "potentially blocking operation", that is, essentially anything that can take time, is forbidden in a protected operation, ensuring that the extent in time of mutual exclusion is kept short. For example, you cannot call an entry of some protected object r2 while you are executing a protected operation on the object r1, which would be the Ada equivalent of nested synchronized blocks.

Condition synchronization requires an entry with a barrier condition. The only way to control access to a shared domain object is by means of a semaphore object similar to the Forklift class in Java. A Forklift protected object would have an entry Acquire with a barrier such as "not busy" and a procedure Release.

In the case of the printer, if it is undesirable to define protected procedures for each different combination of printer operations, a semaphore object is the only solution permitted in Ada. It would be a protected object My_Printer with the entry Acquire and the procedure Release.

5. Conclusions

Java was not originally intended as a language for systems with high reliability requirements, but its popularity has prompted its use for ever wider sets of applications. The Real-Time Specification for Java removes some of the obstacles associated with garbage collection but retains many pitfalls.

Java is adequate for many kinds of concurrent software, but for critical real-time applications it remains a considerably riskier choice than Ada, which was intended for such applications. This is so because Java lacks safeguards against programming errors that are easily committed by an programmer without a sufficiently deep understanding of concurrency issues. There is a trade off here where Java's popularity and the availability of Java programmers must be weighed against the risk exposure caused by those programmer mistakes the language readily allows.

References

[1] Barnes, J. G. P. (1998) *Programming in Ada95,* 2nd Ed., Addison-Wesley.

[2] Burns, A. and Wellings, A. J. (1998) *Concurrency in Ada,* 2nd Ed., Cambridge University Press.

[3] Lea, D. (2000) *Concurrent Programming in Java,* 2nd Ed., Addison-Wesley.

[4] van der Linden, P. (2001) *Just Java 2,* 5th Ed., Prentice Hall.

[5] Holub, A. I. (2000) *Taming Java Threads*, Apress.

[6] Hyde, P. (1999) *Java Thread Programming*, Sams Publishing.

[7] Oaks, S. and Wong, H. (1997) *Java Threads*, O'Reilly.

[8] Brinch Hansen, P. (1999). Java's insecure parallelism, *ACM SIGPLAN* ***Notices*** 34/4**, 38-45**.

[9] Alexander, R. T. and Bieman, J. M. and Viega, J. (2000) Coping with Java programming stress, *IEEE Computer* **33/4**, 30-38

[10] Bollella, G. and Gosling, J. (2000) The real-time specification for Java. *IEEE Computer* **33/6**, 47-54

[11] Bollella, G. and Gosling, J. and Dibble, B. P. and Furr, S. and Turnbull, M. (2000) *The Real-time Specification for Java,* Addison-Wesley.

[12] Brosgol, B. M. (1998) A comparison of the concurrency features of Ada 95 and Java, *Proc. SIGAda '98*, (*Ada Letters* **XVIII/6**,175-192).

[13] Sebesta, R. W. (2002) *Concepts of Programming Languages*, 5th Ed., Addison-Wesley.

[14] Sandén, B. I. (1994). Software Systems Construction with Examples in Ada. Prentice-Hall.

[15] Wellings, A. J. and Johnson, R. W. and Sandén, B. I. and Kienzle, J. and Wolf, T. and Michell, S. (2000) Integrating object-oriented programming and protected objects in Ada 95. ACM TOPLAS 22/3, 506-539. (Reprinted in Ada Letters XXII/2 (June 2002), 11-44.

[16] International J Consortium (2000), Specification, Real-Time Core Extensions, Draft 1.0.14, 2 September 2002. http://www.j-consortium.org

[17] Sandén, B. I. (1996) Using tasks to capture problem concurrency. Ada User Journal 17/1, 25-36.

[18] Vermeulen, A. and Ambler, S. W. and Bumgardner, G. and Metz, E. and Misfeldt, T. and Shur, J. and Thompson, P. (2000) The Elements of Java Style, Cambridge University Press.

[19] Sandén, B. I. (1997) Modeling concurrent software. *IEEE Software* **14/5**, 93-100.

[20] Carter, J. R. and Sandén, B. I. (1998) Practical uses of Ada-95 concurrency features. *IEEE Concurrency* **6/4**, 47-56.

Transition of a Large Project from Ada 83 to Ada 95

Jeff Cousins

BAE SYSTEMS (Combat and Radar Systems) Limited, Apex Tower, 7 High Street, New Malden, Surrey, KT3 4LH
E-mail: jeff.cousins@baesystems.com

Abstract

The Combat and Radar Systems sector of BAE SYSTEMS has successfully produced several large submarine Command and Control Systems for the Royal Navy using Ada 83. Every major release has been delivered on time for the last eight years.

As a prelude to porting these to Ada 95, extensive investigations into Ada 83 to Ada 95 porting issues were performed.

A minimum change approach was desired, with the constraint that wherever possible changes should be backward compatible with Ada 83. Some already delivered systems use hardware that does not have an Ada 95 compiler available, but will have to be maintained for many years.

This paper describes which problems were the most frequent, whether the changes were backward compatible with Ada 83, and which problems had been unexpected.

Most frequent was the long expected change whereby Numeric_Error becomes a renaming of Constraint_Error, followed by package bodies becoming illegal if not required. Unexpected changes included those due to the stricter implementation advice for packed arrays and the stricter rules for Unchecked_Conversion.

Despite the unexpected problems, we have successfully ported systems containing over a million lines of code.

1 Introduction

This article reports on the porting of large submarine Command and Control systems from Ada 83 to Ada 95.

The initial investigation was performed by passing all of the code of the Operational Trainer variant of a submarine Command and Control system through an Ada 95 compiler. The total source code came to over a million lines of Ada.

The "Ada Compatibility Guide" [1] describes many issues, and has even been quoted in requirements specifications. This tends to assume a starting point of pure Ada 83, as subsequently clarified. More likely, the Ada 83 compiler will have followed a face value interpretation of the standard, with vendor extensions to handle the more obvious deficiencies in Ada 83, such as interfacing to other languages, maths library support and interrupt handling.

Most problems were found by simply compiling vast amounts of code.
Binding, linking and executing did reveal further problems, but these were mainly portability problems with our code rather than Ada 95 problems. For example, extra "**pragma** Elaborate"s to ensure that the elaboration order was correct, it being fortuitous that our previous compilers had generated a valid elaboration order. Putting the code through another compiler was a useful exercise in improving code quality even if we had stuck with Ada 83.

Although by no means a trivial exercise, the problems that occurred were small compared with a previous port between big-endian and little-endian target processors.

Where changes were necessary, they were usually simple changes that could be made in a way acceptable to both Ada 83 and Ada 95. For example having a common exception handler for both Numeric_Error and Constraint_Error, or pragmas that are ignored if unrecognised.

Two compilers were used, GNAT and Aonix ObjectAda. All of the code was put through GNAT and 130000 lines were put through ObjectAda for comparison.

Issues are presented in section 2, sorted so as to present the issues that affected the most files first. Where possible, issues identified are listed under the same headings as used in the "Ada Compatibility Guide".

2 Ada 95 Porting Problems

2.1 Numeric_Error renames Constraint_Error

Numeric_Error is now just a renaming of Constraint_Error. Therefore the exception handler for Numeric_Error was removed and the exception handler for Constraint_Error changed from:

```
when CONSTRAINT_ERROR =>
```

to:

```
when CONSTRAINT_ERROR | NUMERIC_ERROR
=>
```

This gave backward compatibility so that when compiled with an Ada 83 compiler a Numeric_Error would be caught by the Constraint_Error handler.

This change had long been anticipated, but until Ada 95 compilers started providing more than minimal Exception_Information the loss of the distinction would have been a hindrance to debugging.

258 files were affected.

2.2 Library Package Bodies Illegal if not Required

Ada 95 reports an error if a body is provided but not demanded by the spec. Initially, "**pragma** Elaborate_Body" was added to the specs, which avoided having to change the bodies.
Under LRM:10.2.1 this meant that the body was elaborated immediately after its declaration though, and in one case this resulted in an incorrect elaboration order.
It was thus decided that it was safer to add a dummy procedure to the spec and body in all cases. For example:

```
package MY_PACKAGE is
  NUMBER_OF_CARDS_IN_NODE : NATURAL;
end MY_PACKAGE;

package body MY_PACKAGE is
begin
  NUMBER_OF_CARDS_IN_NODE :=
   SOME_FUNCTION_TO_INTERROGATE_
   NODE_CALLED_AT_STARTUP;
end MY_PACKAGE;
```

would have been changed to:

```
package MY_PACKAGE is
  NUMBER_OF_CARDS_IN_NODE : NATURAL;
  procedure DUMMY_PROC;
end MY_PACKAGE;

package body MY_PACKAGE is
  procedure DUMMY_PROC is
  begin
    null;
  end DUMMY_PROC;
begin
  NUMBER_OF_CARDS_IN_NODE :=
   SOME_FUNCTION_TO_INTERROGATE_
   NODE_CALLED_AT_STARTUP;
end MY_PACKAGE;
```

79 files were affected.

2.3 Stricter interpretation of implementation advice re packed arrays

With our Ada 83 compilers a size clause can be used to force packing of an array, indeed their manuals confirm this behaviour. This was preferred to "**pragma** Pack" since a compiler is at liberty to ignore a pragma, and different compilers may pack differently.

But the Ada 95 LRM:13.3(53) says:

A Size clause on a composite subtype should not affect the internal layout of components.

This could cause the compiler to warn that the size was too small. Therefore a "**pragma** Pack" was inserted between the array type declaration and the size clause. For example:

```
type BIT_ARRAY_INDEX_TYPE is range 1
 .. N;
type BIT_ARRAY_TYPE is array
 (BIT_ARRAY_INDEX_TYPE) of BOOLEAN;
for BIT_ARRAY_TYPE'Size use N;
```

would have been changed to:

```
type BIT_ARRAY_INDEX_TYPE is range 1
 .. N;
type BIT_ARRAY_TYPE is array
 (BIT_ARRAY_INDEX_TYPE) of BOOLEAN;
pragma PACK (BIT_ARRAY_TYPE);
for BIT_ARRAY_TYPE'Size use N;
```

In some cases the size clause had also to be amended to round it up to a multiple of Storage_Unit, e.g.:

```
for BIT_ARRAY_TYPE'Size use (
 (BIT_ARRAY_INDEX_TYPE'Pos (
 BIT_ARRAY_INDEX_TYPE'Last) /
 System.Storage_Unit) + 1) *
 System.Storage_Unit;
```

71 files were affected.

2.4 Address clauses

The use of address (“**at**”) clauses to achieve overlays was strictly regarded as erroneous by Ada 83 but never the less worked. (Where overlays were used they were for low-level interfacing to external devices). Ada 95 allows it, but the use of an address clause can cause default initialisation to overwrite the original data. This default initialisation may be automatically generated by the compiler and not obvious from the source code of the data types involved.

The Annotated Ada Reference Manual 13.3(12.c) states:

*If the Address of an object is specified, any explicit or implicit initialization takes place as usual, unless a "**pragma** Import" is also specified for the object (in which case any necessary initialization is presumably done in the foreign language).*

Therefore pragmas of the form "pragma Import (Ada, Name_Of_Overlay)" were added.

36 files were affected.

2.5 Vendor-specific Pragmas Removed

1) "**pragma** Interface_Information" was no longer accepted and was changed to "**pragma** Import".

23 files were affected.

2) "**pragma** Export" had a different definition and needed changing to add the calling convention.

1 file was affected.

3) "**pragma** Preserve_Layout" to prevent record re-ordering was no longer accepted and was removed since no

longer necessary with either of the two Ada 95 compilers under investigation.

2 files were affected.

2.6 Obsolescent Features

The use of 'Storage_Size on task types, to control the stack size of tasks, is now obsolescent, and may cause the compiler to give a warning.

Instead "**pragma** Storage_Size" should be inserted within the task specs, though this is not backwardly compatibility with Ada 83.

25 files were affected.

2.7 Vendor-Specific Packages no longer provided

This was one of the few areas where the fixes for Ada 95 were not backwardly compatible with Ada 83, though equivalent packages are now often defined by Ada 95, which will improve portability in the long term and reduce the need for such vendor-specific packages.

1) The vendor specific Math_Library was not provided so the project's maths library was rewritten to make use of the Ada 95 defined Numerics Package. In most cases similar functions were provided, though not factorial, round or truncate. Ada 95 does provide ‘Round and ‘Truncation attributes.

The Ada 95 defined function Arctan raises an exception if both parameters are zero, whereas the Ada 83 libraries had all returned 0. A specific test was added to the project’s maths library to check whether both parameters were zero and if so to return zero.

8 files were affected.

2) The vendor specific package for controlling the mapping of Ada tasks to operating system processes was not provided. References were removed as the Ada 95 compiler provided a sensible default mapping.

1 file was affected.

3) The vendor specific package for defining the source of an interrupt was replaced by Ada.Interrupts.

1 file was affected.

4) The vendor specific package for machine code inserts was not provided. LRM Annex C section 1 provides an alternative, where supported.

1 file was affected. (Where a machine code insert was used it was for low-level interfacing to an external device).

5) A vendor specific Unix types package was not provided. It proved easy to use other types with identical definitions.

6 files were affected.

6) The vendor specific pre-instantiation of Text_IO.Integer_IO was not provided. The Ada 95 package Ada.Integer_Text_IO was used instead.

1 file was affected.

2.8 Stricter rules for Unchecked_Conversion

1) The Ada 95 LRM Section 13.9 (4)-(11) imposes different and additional rules as to whether an Unchecked_Conversion is valid than the Ada 83 LRM Section 3.10.2 did.

Whereas the Ada 83 LRM Section 13.10.2 talks of "*sizes of **objects** of the source and target type*", Ada 95 LRM Section 13.9 (4) explicitly says "*The size of the formal parameter S in an instance of Unchecked_Conversion is that of its **subtype***".

A number of different changes were needed to overcome this problem:

In several places size clauses were applied to type definitions to make the size of the type the same as the size of objects of the type.

In some places the Unchecked_Conversions were unnecessary, both the source and the target being derived from the same type.

Some Unchecked_Conversions between scalars gave unequal size warnings but were proven to give the intended results, so they were left unchanged.

For one Unchecked_Conversion, explicitly converting the source to the parent type, which was of the same length as the target type, was necessary to obtain the intended results.

One Unchecked_Conversion was from a record and not a sensible thing to do since potentially compilers can re-order records; fortunately this was just for information of secondary importance in a fault message that should never occur.

One Unchecked_Conversion from a fixed-point type only gave problems at the extremes of the range.

One Unchecked_Conversion was never used so was deleted.

16 files were affected.

2.9 Real attributes removed

A number of Ada 83 real attributes are no longer provided by Ada 95. In the absence of any readily available guidance in this area, the following mapping was used:

Ada 83	Ada 95	Ada 95 equivalent also in Ada 83
'Mantissa	'Machine_Mantissa	Yes
'Epsilon	'Model_Epsilon	No
'Large	'Last	Yes
'Safe_Small	'Model_Small	No

13 files were affected.

Note that the GNAT compiler still provided the Ada 83 real attributes even though it not in the Ada 95 LRM, but to ensure portability with other Ada 95 compilers this should not be relied on.

2.10 Bad Pragmas Illegal

Bad pragmas are now flagged as errors rather than ignored. Some code had "**pragma** Inline" without saying what to inline. The pragmas were removed as it was no longer intended to inline the particular subprograms, though the obvious alternative would have been to add the parameter saying which subprogram to inline.

10 files were affected.

2.11 System address no longer equivalent to an access type

1)

```
  variable : System.Address := null;
```

was no longer accepted and was changed to

```
  variable : System.Address :=
   System.Null_Address;
```

This was not backwardly compatible with Ada 83.

2 files were affected.

2) Data was passed to and from the lower level communications mechanisms using a generalised data pointer type, called Data_Ptr_Type, and Unchecked_Conversion used to convert between this and an access type pointing to the correct record structure.
This Data_Ptr_Type was defined in terms of Our_System.Address, where package Our_System tried to encapsulate the definition and operations upon Address.
Unfortunately Our_System.Address had been defined as the biggest Integer (System.Min_Int .. System.Max_Int), which is not necessary the same size as the target processor's address range.

As all our target processors were 32 bits, Our_System.Address was re-defined as "**subtype** Our_Types.Integer_4", where Integer_4 had a number of compiler specific definitions so as to always be a 4 byte Integer. Renaming was used added to give visibility of the operations.

Alternative solutions would be to use:

Type System.Storage_Elements.Integer_Address, and conversion functions System.Storage_Elements.To_Address and System.Storage_Elements.To_Integer to convert between this and System.Address. Future 64 bit systems will use this.

Generic package System.Address_To_Access_Conversions. This appeared rather cumbersome though, and would have given a proliferation of instantiations of the generic package.

2 files were affected.

2.12 Unconstrained Generic Actual Subtypes

A generic can no longer be instantiated with an unconstrained actual subtype to meet a formal private type unless the spec of the generic explicitly indicates that it is allowed to be unconstrained.

The spec was of the form:

```
generic
  ... -- Some other declarations
  type TABLE_TYPE is limited private;
  type INDEX_TYPE is (<>);
  ... -- Some other declarations
package GEN_PACKAGE is
```

This was changed to:

```
generic
  ... -- Some other declarations
  type INDEX_TYPE is (<>);
  type ITEM_TYPE  is limited private;
  type TABLE_TYPE is array (INDEX_TYPE
   range <> ) of ITEM_TYPE;
  ... -- Some other declarations
package GEN_PACKAGE is
```

to be similar to the example in chapter 13.2 of Barnes [2].

3 files were affected.

2.13 Compatibility Checks at Compile-Time

When instantiating a generic, actual and formal array types must both be constrained or both be unconstrained (LRM:12.5.3(5)).
In our case the generic parameter was changed from an unconstrained array to a constrained array, since the package was only ever instantiated with constrained arrays.

2 files were affected.

2.14 Vendor-Specific System Extension Removed

1) Our Ada 83 compilers provided a "hook" in package System that could be called from exception handlers to force output of the call chain.
This is not in the Ada 95 LRM, resulting in an error.

In the short term, the call was commented out.

Subsequently GNAT provided the call chain as part of the Exception_Information in package Ada.Exceptions. The GNAT extension GNAT.Traceback.Symbolic was also used to provide this information symbolically.
There was a problem though that Exception_Information needed an Exception_Occurrence parameter, the Exception_Occurrence being a label attached to the exception handler, which was not backwardly compatible with Ada 83.
The GNAT extension GNAT.Most_Recent_Exception was used to provide similar functionality to the Ada 83 vendor extension.

Although we have tried hard to avoid vendor extensions, the GNAT extensions in the exception handling area were regarded as indispensable.

1 file was affected.

3 Subsequent Development

Copies of the GNAT, Aonix and Rational Ada 95 compilers were received for evaluation. Only GNAT could handle all of our code without failing with internal errors.

On the rare occasions where we had a problem with GNAT, an acknowledgement would be received by return of e-mail. A fix, or an explanation of what we were doing wrong, would be received within a day or two.

This was a radically different relationship from that which we had had with our Ada 83 compiler vendors. Previously it would take from 3 to 6 months to receive an acknowledgement for a bug report, and if a fix was really important to us we might get it in the maintenance release at the end of the following year.

A couple of surprises with GNAT were that numeric overflow checking and stack overflow checking were not provided by default, but required compiler switches to enable them. A compiler switch was also required to enable dynamic elaboration order checks, static elaboration order checking was the default but would have required far too many changes to our legacy code.

Level 2 optimisation was required to obtain code of comparable efficiency to that which our Ada 83 compilers had provided by default, and appears to be a de facto standard amongst GNAT users. The image file sizes were twice the size of those produced by our Ada 83 compilers but the sizes resident in memory were similar.

New code had some opportunity to take advantage of the new features of Ada 95, but these proved to be of only limited value in practice. The biggest change of Ada 95 was probably the introduction of tagged types for object-oriented programming. Although our design is quite object-oriented, tagged types have only found local use, within the latest release of our infrastructure or middle-ware.

Our systems are distributed and heterogeneous, with messages flying back and forth between over a hundred processors. The message types are realised as the discriminants of variant records, with representation clauses being used to enforce a mutual understanding across all the processors.

Run-time dispatching depending on the tag did initially look attractive. Unfortunately tags are typically implemented as the address of a jump table, so their values do not have meaning on another processor with a different address space, and cannot be controlled by representation clauses. Tags can be inter-worked between processors by converting to and from their "external tag", represented as a string. This is the method adopted by GLADE (GNAT Library for Ada Distributed Execution) [3], but the overhead would have been unacceptable.

Some of the smaller changes have been beneficial for interfacing. The access parameter mode maps well on to the pointers of imported c routines, and allows the compiler to check that the parameter accesses an object of the expected type, rather than just passing a System.Address.

Modular types allow easier bit manipulation when interfacing to hardware. Ada 83 on '80s or early '90s processors in real-time systems almost invariably required machine code inserts for performance reasons.

4 The future

Our "wish list" for what we would like to see in Ada 0Y is quite modest:

1) A simple way of triggering the output of a symbolic call-chain for the most recent exception.

2) We would like to be able to perform Unchecked_Conversion between any objects of the same size, even if their types are of different sizes.

3) Sometimes a generic body can meet the contract of its spec but can only be legally instantiated at library level, not at a lower level. This is because of the rules on downward closure for access to subprogram types, to avoid dangling pointers.
A solution to this is desirable, possibly by adding run-time accessibility checks for access to subprogram types. This is being investigated by WG9 under AI95-254.

4) It is a bounded error to invoke a potentially blocking operation within a protected action. A Program_Error is only raised if the bounded error is "detected". Currently a compiler can avoid raising the Program_Error by the vendor arguing that the bounded error is never detected, even in the most obvious cases. The concept of detection needs to be clarified.

5) It would be useful for the private part of a public package to be able to “**with**” private siblings. This is being investigated by WG9 under AI95-262.

References

[1] Bill Taylor (1995), Ada Compatibility Guide, Version 6.0, Transition Technology Limited.

[2] J G P Barnes (1994), Programming in Ada, Fourth Edition, Addison-Wesley.

[3] Josef Aichorn (2000), Experimental Performance Analysis of GLADE's *Implementation of the Distributed Ada95 Programming Model*, Ada User Journal, vol 21, no 3.

Ada UK 2002 Sponsors

ACT Europe
Contact: Franco Gasperoni
8, Rue de Milan, 75009, Paris, France
Tel: +33-1-49-70-67-16 Fax: +33-1-49-70-05-52
Email: sales@act-europe.fr URL: www.act-europe.fr

Alenia Marconi Systems
Contact: Don Harvey
Eastwood House, Glebe Rd., Chelmsford, Essex, CM1 1QW, UK
Tel: +44-(0)1276-696901 Fax: +44-(0)1276-659842
Email: don.harvey@amsjv.com URL: www.aleniamarconisystems.com

Aonix Europe Ltd
Contact: Neil Michniak
Partridge House, Newtown Rd., Henley on Thames, Oxon, RG9 1HG, UK
Tel: +44-(0)14941-415000 Fax: +44-(0)14941-571866
Email info@aonix.co.uk URL: www.aonix.com

ARTiSAN Software Tools
Contact: Peter Kibble
Stamford House, Regent St., Cheltenham, Glos., GL50 1HN, UK
Tel: +44-(0)1242-229320 Fax: +44-(0)1242-229301
Email: peterk@artisansw.com URL: www.artisansw.com

BAE SYSTEMS
Contact: Paul McCormack
Warwick House, PO Box 87, Farnborough Aerospace Centre, Farnborough, Hants, GU14 6YU, UK
Email: Paul.McCormack@baesystems.com URL: www.baesystems.com

Data Systems and Solutions
Contact: Dave Woodhall
SEAS Building, Sinfin Lane, Derby, DE24 8BJ, UK
Tel: +44-(0)1332-771700 Fax: +44-(0)1332-770921
Email: info@ds-s.com URL: www.ds-s.com

EDS
Contact: Lee Edwards
Hartley House, 15 Bartley Wood Business Park, Bartley Way, Hook, Hants., RG27 9XA, UK
Tel: +44-(0)1256-741122 Fax: +44-(0)1256-741132
Email: swep.sales@eds.com

First Matrix Ltd
Contact: Alan Barker
Old Lion Court, High St,. Marlborough, Wilts., SN8 1HQ., UK
Tel: +44-(0)1672-515510 Fax: +44-(0)1672-515514
Email: arb@ftmx.com

Green Hills Software Ltd
Contact: Jon Williams
Goodsons Mews, Wellington Street, Thame, Oxon, OX9 3BX, UK
Tel: +44-(0)1844-267950 Fax: +44-(0)1844-267955
Email: sales-uk@ghs.com URL: www.ghs.com

IPL Information Processing Ltd
Contact: Ian Gilchrist
Eveleigh House, Grove St., Bath, BA1 5R., UK
Tel: +44-(0)1225-475114 Fax: +44-(0)1225-444400
Email: ipl@iplbath.com URL: www.iplbath.com

LDRA Ltd
Contact: Jim Kelly
24 Newtown Rd., Newbury, Berks., RG14 7BN, UK
Tel: +44-(0)635-528828 Fax: +44-(0)635-528657
Email: sales@ldra.com URL: www.ldra.com

Objektum
Contact: Derek Russell or Ahmed Amin
Units 2/3 Cranleigh Works, The Common, Cranleigh, GU6 8SB, UK
Tel: +44-(0)1483-278178 Fax: +44-(0)1483-275384
Email: info@objektum.com URL: www.objektum.com

Praxis Critical Systems Ltd
Contact: Peter Amey
20 Manvers St., Bath, BA1 1PX, UK
Tel: +44-(0)1225-469991 Fax: +44-(0)1225-469006
Email: sparkinfo@praxis-cs.co.uk URL: www.praxis-cs.co.uk

Rational Software Ltd
Contact: Roger Bowser
Kingswood, Kings Ride, Ascot, Berks., SL5 8AJ, UK
Tel: +44-(0)1344-295000 Fax: +44-(0)1344-295001
Email: info@rational.com URL: www.rational.com

John Robinson & Associates
Contact: John Robinson
2 Currer St., Oakenshaw, Bradford, W. Yorks., BD12 7DP, UK
Tel: +44-(0)1274-691935 Fax: +44-(0)8700-558750
Email: John@jr-and-assoc.demon.co.uk URL: www.jr-and-assoc.demon.co.uk

Telelogic UK Ltd
Contact:
Chancery House, 8 Edward St., Birmingham, B1 2RX, UK
Tel: +44-(0)121-2346600 Fax: +44-(0)121-2346611
Email: info@telelogic.com URL: www.telelogic.com

TNI Europe Ltd
Contact: Tony Elliston
58a Mill St., Congleton, Cheshire, CW12 1AG, UK
Tel: +44-(0)1260-291449 Fax: +44-(0)1260-291449
Email: info@tni-europe.com URL: www.tni-europe.com

Wind River Systems UK Ltd
Contact: David Bew
Unit 5 & 6, 1st Floor, Ashted Lock Way, Aston Science Park, Birmingham, B7 4AZ, UK
Tel: +44-(0)121-3590999 Fax: +44-(0)121-3804444
Email: inquiries-uk@windriver.com URL: www.windriver.com