ADA USER JOURNAL

Volume 26
Number 1
March 2005

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 license to publish (and, if appropriate, republish) it, if and when the article is accepted for publication. We do not require that authors assign copyright to the Journal.

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

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, preferably in electronic format. The Editor will only accept typed manuscripts 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

You will recall that the closing issue of the previous Volume of Journal contained the first installment in a series of articles that will eventually compose the new Rationale document for Ada 2005. We plan to continue the publication of the successive installments throughout the whole of Volume 26. We also plan, in the New Year, to publish a special issue of the Journal, which will bundle the whole Rationale document. That should make a valuable gift to our readership! The installment of the Rationale that we singled out for this issue describes the various (and numerous!) improvements that Ada 2005 makes to the object-oriented model of the language. For as much as I'd like, it is not for the editorial to enter the technicalities of this topic: the 18-page article by John Barnes is there to give you the full picture of it. Of course, much appreciation information value comes to this issue also from the usual News and Calendar sections that are prepared with great dedication by the respective editors, Santiago Urueña and Dirk Craeynest. That's all there is to say for now. Enjoy the reading (and let us know you do)!

Tullio Vardanega
Padova
March 2005
Email: tullio.vardanega@math.unipd.it

News

Santiago Urueña

Technical University of Madrid. Email: suruena@datsi.fi.upm.es

Contents

Ada-related Events

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

July 6-10 - 2004 Libre Software Meeting Ada Presentations

From: Lionel Draghi <Lionel.Draghi@Ada-France.org>
Date: Fri, 26 Nov 2004 00:07:05 +0100
Subject: présentations RMLL 2004 dispos (presentations Libre Software Meeting 2004 are available)
Newsgroups: fr.comp.lang.ada

[Translated from French – su] The presentations given during the Ada session in the "Rencontre Mondiale du Logiciel Libre (RMLL" are available at the address: http://www.ada-france.org/article115.html. They are in English and/or French. Many thanks to the speakers and now see you all at FOSDEM.

I take this opportunity to remind everyone that the Ada-France site is open to all contributions. Unlike Quentin Ochem, everyone else should not have unbounded documents to publish (http://www.ada-france.org/article116.html).

To any rate: if you publish a short note on the latest XPath library in Ada, that will allow all who do not follow comp.lang.ada to become aware that that exists. And every time that anyone will search for the term XML, he/she will get to your note (as it occurs here at http://www.ada-france.org/mot32.html?var_recherche=xml). Overall, that will have been a useful quarter of an hour.

Happy reading to everyone.

[See also "Libre Software Meeting" in AUJ 25-3 (Sep 2004), p.117. -- su]

Jun 20-24 - Ada-Europe 2005 Conference

From: Dirk Craeynest <dirk@heli.cs.kuleuven.ac.be>
Organization: Ada-Europe, c/o Dept. of Computer Science, K.U.Leuven
Date: 4 Jan 2005 23:54:00 +0100
Subject: FINAL CfIP, Conference Reliable Software Technologies, Ada-Europe 2005
Newsgroups: comp.lang.ada,fr.comp.lang.ada

Final Call For Industrial Presentations

10th International Conference on Reliable Software Technologies - Ada-Europe 2005

20 - 24 June 2005, York, UK

http://www.ada-europe.org/conference2005.html

* DEADLINE Monday 10 JANUARY *

For more information please see the conference Web site and select "Call for Industrial Presentations".

The 10th International Conference on Reliable Software Technologies (Ada-Europe 2005) will take place in York, UK. Following the usual style, the conference will span a full week, including a three-day technical program and vendor exhibitions from Tuesday to Thursday, along with parallel workshops and tutorials on Monday and Friday.

In addition to the usual call for papers, this year we are also having a call for presentations primarily aimed at industrialists who have valuable experience to report but who do not wish to write a complete paper.

This separate call for presentations is made for Experience Reports from Industrial Projects and/or Experiments, Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics, Experience Reports on Education and Training Activities with bearing on any of the conference topics. See the conference web site for further details.

Presenters are invited to submit a one-page overview of the proposed presentation to Rod Chapman (rod.chapman@praxis-his.com) by January 10th 2005. The Industrial Committee will review the proposals. The authors of selected presentations shall prepare their final presentation by 20th May 2005; they should aim to talk for 20 minutes. The authors of accepted presentations will also be asked to derive articles from them, for publication in the Ada User Journal.

Schedule:

10 January 2005: Submission of one-page overview

17 January 2005: Notification to authors

20 May 2005: Presentation required

20-24 June 2005: Conference

[See also same topic in AUJ 25-4 (Dec 2004) pp.181-183 -- su]

Ada-related Resources

6th Birthday of AdaPower.com

From: David Botton <david@botton.com>
Date: Sat, 6 Nov 2004 23:45:52 -0500
Subject: Happy B-Day Ada Power
Newsgroups: comp.lang.ada

November 7, 2004 - Is the 6th Birthday of AdaPower.com!

If you haven't had a chance to see the new database driven design and contents (still more sitting in my queue to be put in over the next week). Now would be the perfect time □

[See also "New AdaPower.com" in AUJ 25-4 (Dec 2004), p.184. -- su]

Searching all known Ada sites

From: David Botton <david@botton.com>
Subject: AdaPower Search
Date: Mon, 1 Nov 2004 00:55:54 -0500
Newsgroups: comp.lang.ada

The new AdaPower is quickly being filled up with the old contents and the new (more than 250 links and articles already in and more to come - I expect to have everything imported and up to date by end of week)!

So how do you find what you are looking for?

Simple, AdaPower now features a search box off the front page that will instantly help you find the Package for reuse, code example or more that you are looking for.

The new AdaPower.com - more power, more Ada, more coding excellence!

[See also "New AdaPower.com" in AUJ 25-4 (Dec 2004), p.184. -- su]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 1 Nov 2004 17:54:59 -0600
Subject: Re: AdaPower Search
Newsgroups: comp.lang.ada

Of course, if you want to search all known Ada sites, including AdaPower (and avoid all of the non-Ada stuff that pops up on general search engines), you can use the Ada-wide search engine at http://www.adaic.com/site/wide-search.html.

Randy Brukardt, Technical Webmaster, adaic.com/.org

[See also "AdaIC Opens Ada Sites Search Engine" in AUJ 24-2 (Jun 2003), pp.72-73 -- su]

AdaWorld.com and AdaPower.com

From: David Botton <david@botton.com>
Date: Wed, 3 Nov 2004 23:22:08 -0500
Subject: AdaWorld and AdaPower
Newsgroups: comp.lang.ada

(Stephane, not sure if this is the best place to discuss this, so I am comfortable on the admin-list @ AdaPower.com or in private e-mails if you prefer).

Some time back we got in to some discussion, but never really focused on the different needs and roles our sites could serve to the community.

AdaPower has always had its main focus as being a practical "tool" for Ada the language (it started as the Ada Source Code Treasury off my home page and grew in to AdaPower), but not really on the Ada community (sure it does a bit of that, but has never been "the" site for that). I don't foresee that changing and in fact with the new design and some future plans, I see it becoming even more focused on code, tutorials, articles, etc.

In fact if I was going to do another site, I would add an Ada community center, I would create something say called AdaWorld :-)

AdaWorld has already made strong head way in this regard. I would think that we put our heads together on AdaWorld, much as you have done so for AdaPower in the past and now, and push forward head strong in to it (I am ready to put my money behind my mouth here so to speak an be part of making this happen if you would like also) and make it both the compliment and to some degree the umbrella in relationship to AdaPower and other like sites.

Here is how I think AdaWorld can and should become the Ada Community Site (perhaps you or others see more):

* Current events that relate to the Ada community

- Conferences / Call to papers

A fantastic list is currently kept up to date by Dirk Craeynest at http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/index.html. That list could be mirrored at AdaWorld and reach more people.

- Product Announcements

Perhaps a form that would output press releases and announcements for Ada product releases, releases of new versions of code packages for reuse, etc. It could automatically send out a message to CLA, a list that people could join off of AdaWorld, and perhaps Team-Ada. I realize that there is a list at Ada IC, but very sadly it is focused on supporting ARA vendors and not the Ada community as a whole (although certainly this does so on many levels, but not in the capacity at hand). An AdaWorld open alternative for Ada PR stories / announcements would be a welcome addition to the Ada World :-)

- Ada course announcements

There are courses being given on Ada even for free in various places.

For example, I am considering doing a series of live web cast tutorials on Ada. Getting some others to do the same. A good community center would be key to making this type of work a success.

* Ada Advocacy (in general and for the common man)

A center for collecting together much as AdaPower does for code and packages, of advocacy information. There is tons of it spread thin all over the net. While there are some Ada advocacy sites, they are fairly centered on certain themes. Some oriented around dependability, others coding readability, etc. etc. They also target varied markets. One market not being focused on and should be a big part of this is the common man, the application developer and the IT dude. The Big Linux book goes a long way to reach out to the common man, but a solid resource for Ada advocacy to non-critical engineering types is _very badly_ needed!

* Community Guide (as a guided tour)

A step by step guide in a "tour" book format to getting started in the Ada community.

- Where to go for code for reuse.

- Where to talk about Ada and get help.

- The history of Ada

- Where Ada is going and where it has been

- Key things to try out in Ada that will make you want to stick with the language

And much more.

* Cool Factor Factory

Ada needs to be cool. The very thing that has "killed" Ada can be what makes it the biggest "turn on". When I find a good geeky high schooler / CS1er around that I want to convert to Ada, I tell him using Ada via GNATCOM you can program missiles to fire from a word document (and that is a fact!). I tell him stuff like:

Ada - Military Grade Programming!

If you can get it to compile man, you just about know its goin' to work. Check this out (examples shown)

I am not going to sell the M$ generation on reality, damn M$ already made us believe bugs are features too :-) Vote for, Bug rights now!

If I could see straight (its getting late), I'd write more, but I think the picture is clear as to the need for an Ada community site. AdaWorld has already started to dance in that space, the other sites that exist are too focused and/or not maintained. I think and would want to be part of an AdaWorld taking the lead in this space.

My 2 dollars and 43 cents :-)

David Botton, http://www.adapower.com

From: Stephane Richard <stephane.richard@verizon.net>
Date: Thu, 04 Nov 2004 11:28:26 GMT
Subject: Re: AdaWorld and AdaPower
Newsgroups: comp.lang.ada

No problem discussing this here at all. :-). If I'm gonna push towards a community website, I might as well talk to the community no? :-). See my comments distributed (logically I hope) in your post.

I remember that, indeed we never went into details but we already seemed to agree that different websites should work together with other websites instead of imitating them. There should be no competition but rather a synergy between existing websites.

When I started Ada World (coincidence? hehe) I wanted it to be somewhat of a community center in a way yes, maybe even a non regular date Ada Magazine (at the time I didn't want to abide to any Volume/Issue dated release :-) to some point too. I wanted people visiting the website to get a good glimpse at what's out there (the actively developed projects as well as basically who's using it and why). So yes a Community / Magazine oriented website (with a bit of _____ fill in the blank to go and get interest from other programmers in other languages).

Two heads are definitely better than one. And I'm always open to suggestions from you and anyone else that would like to see something they're not seeing yet. :-).

[...] A company that recently discovered my website asked me to add their products in a "magazine" kind of way about a week ago. And that got me thinking in that direction. Of course not

just for products, for anything that's going on :-).

[...] I wasn't thinking exactly what to put in, but I was thinking that something like that might be fun. I was about to start asking around get an idea if this kind of thing could have a potential interest. ;-) It just wasn't this detailed in my head yet ;-) *It was late for you when you wrote this, it's way too early for me right now, not enough caffeine running through my veins yet :-).

[...] I haven't been in the Ada community long enough yet to notice what you're saying [about Ada Advocacy]. Other than if I'm looking for non targeted advocacy, I haven't seen too many in my searches. But I won't say I haven't seen any :-).

I think Ada is cool ;-). But yeah, I know what you mean here. I've been saying it for a while, but people have a "software engineering" point of view of Ada. In a way they're right and that's what makes the strength of Ada, readability too of course, and many more things. GNAVI when completed I think will definitely help with the cool part. Being Ada's reply to Delphi, the outside world will have something they know (Delphi) to compare the reply (GNAVI). And I think we need more this in and out of Ada to compare Ada.

[...] There will be daylight tomorrow too, we got time to talk :-). Me I'm still waiting for daylight to start showing itself (early like I said). But yeah, there's room for Ada World to grow. And so far, what you said here, even if it was late last night, fits Ada World's Future plans pretty good. So we can talk about here (maybe get opinions and ideas from people here) to push us in the right direction and well then start walking ;-).

I'm Canadian, so this was my 3 dollars and 69 cents :-).

Stephane Richard ("Ada World" webmaster), http://www.adaworld.com

From: Tom <8f27iw6z@canada.com>
Date: 5 Nov 2004 15:51:34 -0800
Subject: Re: AdaWorld and AdaPower
Newsgroups: comp.lang.ada

May I make a suggestion: Instead of just product announcements is there any way to add reviews to the products? I am thinking of something like an article that compares and contrasts the new product with existing products. What would be also helpful is if there were a way for current users to add their opinions to the end of the article. I am probably dreaming in Technicolor, but could the articles also include the prices of the products?

Along the same line is there a way to get an article written that discusses all the major products in a product field. What I am thinking is an article that compares the major strengths and weaknesses for example of the ten most used Ada IDEs (including their compilers). Something like what CNET does when it compares groups of products. User feedback would also be useful here.

If you did both the reviews of all the products and major evolution it would very helpful in showing the new comers how the products are actually evaluated. Another benefit would be to make it easier to find the lesser known but great products; because, they have a great possibility of being mentioned by users who are giving their opinions about the reviewed products.

[For the section "Community Guide"] I have a suggestion also: How about articles by developers in the field? For instance, I would find it very interesting to see what programmers in the scientific and engineering arenas have to say about products needed or desired for doing numerical programming. Why were the particular IDEs and other programs used or desired. Also I would like to see the opinions of the readers of the articles included.

From: Stephane Richard
<stephane.richard@verizon.net>
Date: Sat, 06 Nov 2004 00:41:01
Subject: Re: AdaWorld and AdaPower
Newsgroups: comp.lang.ada

That's a very interesting suggestion Tom. I can definitely see it happening too. As far as the prices, well I wouldn't mind showing them, but as long as visitors recognize that the prices shown are valid for that time / date the article was written :-) Not sure I'd want to run back through the articles to update prices :-).

Do you mean comparison charts? Columnized data for quick comparison of features/capacities? That will require research, but I know I'd like that (even if it's not what you meant :-) but yeah I see your point of view. As in: "Hey I need something that does this or that, which one offers more/less/fastest etc etc" in a quickly readable format.

I aim at more than just the newcomer. I want to go get the non newcomer too if it can be said that way :-). I want anyone that happens to come visit Ada World to say "hey, what's this all about?," and actually be able to answer that question. I think these two would definitely help in answering these questions and although they would represent research, it's a small investment to make to mark up the quality of what they can find on Ada World, and I'm ready to commit myself to it.

This is not that hard to accomplish, today's most popular CMS (content management systems) already are setup with articles and comments/reviews already in place. I do like the industry specific point of view approach idea. It's definitely noted in big characters (bold, italic and underlined ;-) on my "to consider" list. If I have anything to say about it, it should be there...Wait a minute, it's my website. I do have something to say about it ;-) I hope hehe...but yeah seriously, I like the industry specific areas like that. Definitely.

Those are excellent suggestions and as I write this reply I can tell you that they are already thrown on the discussion table :-). Thank you and feel free to suggest more. That goes for everyone reading this too ☺

From: David Botton <david@botton.com>
Date: Sat, 6 Nov 2004 19:33:28 -0500
Subject: Re: AdaWorld and AdaPower
Newsgroups: comp.lang.ada

I think that is an amazing idea. [...]

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Thu, 4 Nov 2004 14:20:26 -0600
Subject: Re: AdaWorld and AdaPower
Newsgroups: comp.lang.ada

This description of what AdaIC does is not quite correct, so let me explain. The AdaIC announcement list is a *filtered* list of announcements. We try to run only things of wide-spread interest. I know I don't want to see press releases about some company you've never heard of choosing someone's product for a project, and I don't much care that version 3.1.8.7a of something is now available, either. The vendor should use their customer mailing lists to get that sort of information out.

So we only run announcements of *new* Ada-related products, *major* conferences, and the like. Otherwise, the list would fill your mailbox with enough useless junk that you'd start thinking it was spam.

Yes, we do give priority (and relax the filters a bit) for ARA vendor articles. But the real problem is that hardly anyone sends us announcements, and there is only so much that can be scavenged off of comp.lang.ada (and that leads heavily to non-commercial stuff -- we want a balance). I've probably missed a few announcements for new products/bindings/etc. here (thinking that they were just another release, or forgetting about them altogether); but it works better if you send them to us at webmaster@adaic.com.

Another thing you should know is that the powers that be are planning a redesign/refresh of the AdaIC site with the intent of increasing its Ada Advocacy focus. I'm under orders to do as little as possible with the current site (especially not page corrections) in order to not duplicate effort (meaning that there won't be much new content beyond news and jobs for a while). I have no idea what the ultimate result of that redesign/refresh will be.

From: David Botton <david@botton.com>
Date: Wed, 24 Nov 2004 23:24:19 -0500
Subject: About the Ada FAQ

Newsgroups: comp.lang.ada

I will be opening up the Ada FAQ for wiki style editing soon. In the mean time please send complete text (Category/Question/Answer) with corrections or additions to me via the AdaPower contact form or to David@Botton.com

Ada at Wikipedia & Wikibooks

From: Martin Krischik <martin@krischik.com>
Date: Sat, 06 Nov 2004 11:48:26 +0100
Subject: Wiki on Ada
Newsgroups: comp.lang.ada

There are two main entries for Ada in Wiki:

http://en.wiktionary.org/wiki/Ada
http://en.wikibooks.org/wiki/Programming:Ada

But they could do with some improvements. Especially the Wikibooks Entry.

And it is very easy: Just grab a *random* keyword which has no article yet and write something about it. Everybody can do that.

From: Preben Randhol <randhol@bacchus.pvv.ntnu.no>
Date: Mon, 15 Nov 2004 07:34:13 +0000
Subject: Re: Wiki on Ada
Newsgroups: comp.lang.ada

Don't forget wikipedia which is huge:

http://en.wikipedia.org/wiki/Ada_programming_language

http://en.wikipedia.org/wiki/Ada_Lovelace

From: Martin Krischik <martin@krischik.com>
Date: Sun, 07 Nov 2004 09:40:12 +0100
Subject: Re: Wiki on Ada
Newsgroups: comp.lang.ada

Steve wrote:

> After reading the Programming:Ada entry, which has a mix half truths and misconceptions,

Sorry, haven't fixed them. Always feel a bit uneasy on changing other people's work. But I should not - it is Wiki after all.

> it makes me wonder: Does Wikibooks have any sort of reputation for accuracy? If it does, some work needs to be done.

Well I wanted information on say "Eiffel" and came across Wiki on Eiffel I would expect them to be correct. That is because I would expect the Eiffel community and Eiffel advocacy the have written them.

This expectation might be not be correct but that's the way it is. And here lies the problem: The Ada entries have not been done by Ada advocates but just been copied together by an Ada amateur. I checked the other articles done by the original Author and it looks like he is a Wiki supporter just creating Wiki entries for the sake of creation Wiki entries.

We should not leave them like they are. We could of course delete them on the base of being incorrect - But another Wiki Advocate might create new one which would not be helpful.

> If it is generally recognized as half truths and misconceptions, then it is probably not worth correcting.

I think it is always worth fixing it. Wiki is a great change. That's why I wrote the call in the first place. You have half an hour spare? Write an article on Programming:Ada:Operators:*. Got half a quarter more? Write an article on Programming:Ada:Operators:+ including the use of type conversion operator.

Got a lot of time at hand? Programming:Ada:Tasking.

The chance with Wiki is that each of us doesn't need to spend lots of time on it.

A classic Website just needs a lot of time for the Webmaster. And if you are not the Webmaster then your articles might not be accepted. Well, my articles for the classic AdaPower where just ignored.

From: Martin Krischik <martin@krischik.com>
Subject: Re: Wiki on Ada
Date: Sun, 07 Nov 2004 18:26:35 +0100
Newsgroups: comp.lang.ada

Björn Persson wrote:

> I tried to fix the broken link from Wiktionary to Wikibooks, but I couldn't figure out what was wrong with it. Apparently I'd have to study the special wiki language first.

Well, standard http://.... links are automatically recognised. If you need a title you enclose the link in [..] like [http://ada.krischik.com My Ada Page]

Ok, I fixed it - my mistake - used the syntax for wiki internal links [[...]] for an external link.

From: Jacob Sparre Andersen <sparre@nbi.dk>
Date: 08 Nov 2004 11:39:52 +0100
Subject: Re: Wikibooks Programming:Ada:Installing
Newsgroups: comp.lang.ada

Martin Krischik wrote:

> I added the Platforms I know of:
> http://en.wikibooks.org/wiki/Programming:Ada:Installing
> Anything more? Especially Debian is missing!

Not any more. :-)

But it would be popular if somebody would add MacOS X to the list.

From: Martin Krischik <martin@krischik.com>
Date: Wed, 17 Nov 2004 19:50:54 +0100
Subject: Wiki: Need some help from non GNAT users.
Newsgroups: comp.lang.ada

Maybe some of you who don't use GNAT could help me out with http://en.wikibooks.org/wiki/Programming:Ada:Packages:Standard

From: Martin Krischik <martin@krischik.com>
Date: Tue, 23 Nov 2004 13:46:11 +0100
Subject: GWindows in wiki
Newsgroups: comp.lang.ada

Someone had the great Idea to add a http://en.wikibooks.org/wiki/Programming:Ada#Other_Language_Libraries section to Programming:Ada. And - in line with the fact that Programming:Ada is a tutorial and not a link collection - he or she added some demo code for GWindows: http://en.wikibooks.org/wiki/Programming:Ada:Libraries:Database:GWindows

However there are some important information missing. Could somebody with GWindows experience fill in the blanks. Look at http://en.wikibooks.org/wiki/Programming:Ada:Libraries:MultiPurpose:AdaCL http://en.wikibooks.org/wiki/Programming:Ada:Libraries:Container:Booch on what it should look like.

From: Martin Krischik <martin@krischik.com>
Date: Tue, 01 Feb 2005 09:26:02 +0100
Subject: Ada 2005 on wiki
Newsgroups: comp.lang.ada

I like to announce my new Wiki page on Ada 2005: http://en.wikibooks.org/wiki/Programming:Ada:2005

It is intended as an index - the actual features are added right into the main "Programming:Ada" wikibook. I hope that we have a free Ada 2005 tutorial right along with with the Ada 2005 compilers.

Like almost all Wiki-Pages it's work in progress and everybody is invited to contribute.

Koders - Source Code Search Engine

From: Stephane Richard <stephane.richard@verizon.net>
Date: Tue, 09 Nov 2004 19:11:57
Subject: Re: A source code search engine
Newsgroups: comp.lang.ada

Lionel Draghi wrote:

> http://www.koders.com/ is an interesting search engine for free licensed code. Unfortunately, Ada is not yet listed in the 16 supported languages.

And now it's the time we find out if they are open to improvements :-). I just requested the addition of the Ada programming language in their search

facilities. Stated a few sites where they can find source code and emphasized the existence of a considerable pool of Ada source code equivalent to most other languages out there if they "know" how to search (much more politely of course ;-) but we'll see what happens...I'll keep everyone posted here.

From: Szymon Guz <guzo@stud.ics.p.lodz.pl>
Subject: Re: A source code search engine
Date: Wed, 10 Nov 2004 00:13:39
Newsgroups: comp.lang.ada

Maybe it helps a little, but yesterday I requested addition of the Ada too.

From: Stephane Richard <stephane.richard@verizon.net>
Date: Sat, 13 Nov 2004 01:49:27
Subject: Re: A source code search engine
Newsgroups: comp.lang.ada

Just to let you guys know. I've gotten a reply to my request to add Ada to his search engine. Perhaps others got this reply too. But just in case, and/or for those who haven't contacted them yet. I'm including his reply below:

> Hi Stephane,
> Ada support is certainly on the roadmap. We have many languages to add and we're going to try to support as many as possible. As far as a deadline, I'm unable to offer you anything concrete.
> Regards,
> Ankur

So it's just a question of time :-). Soon enough, Ada will be there.

From: Lionel Draghi <Lionel.Draghi@Ada-France.org>
Date: Wed, 12 Jan 2005 21:08:23 +0100
Subject: Re: A source code search engine
Newsgroups: comp.lang.ada

The Koders source code search engine is now Ada enabled:
http://www.koders.com/

Ada IRC channel on Freenode

From: Genro Kane Gupta <genro@niestu.com>
Date: 16 Jan 2005 20:41:18
Subject: [Announce] #Ada IRC channel on Freenode
Newsgroups: comp.lang.ada

This is the annual reminder of the existence of the #Ada channel on the Freenode IRC network. Now entering its fourth year, the channel is open to all discussions related to the Ada language and its use. We welcome beginners and pros alike, and do our best to maintain a friendly, productive, and informative atmosphere.

Point your IRC client to irc.freenode.net and join the #Ada channel. Come one, come all!

[See also "Ada IRC Channel" in AUJ 25-1 (Mar 2004), p.7. -- su]

Ada Conformity Assessment Test Suite Updated

URL: http://www.adaic.org/compilers/acats/2.5/mods/mods2_5l.html
Date: January 7, 2005

ACATS Modification List 2.5L and the associated test files have been posted.

Ada and Education

Ada at www.techtutorials.info

From: Stephane Richard <stephane.richard@verizon.net>
Date: Mon, 15 Nov 2004 13:01:42
Subject: Re: Windows Service (Ada)
Newsgroups: comp.lang.ada

David Amies wrote:

> I prefer to learn by reading manual's and other web pages; however I can't seem to find anything relevant to what I am trying to do. Probably this is just cause I'm new to this language and don't know where to look, so I am hoping for a few pointers to some helpful relevant links.

[...] I would have to suggest brushing up your Ada skills a bit, since it's been a long time since you did C programming. Just take a look here: http://www.techtutorials.info/prada.html, Ada Power has some interesting reading material as well. Also, my website has a learning center, take a look at the tutorials there.

Ada Tutorials for C++ programmers

From: Quentin Ochem
Date: Thu, 25 Nov 2004
Title: Ada pour le programmeur C++
URL: http://www.ada-france.org/article116.html

[Translated from French – su] The attached document presents the Ada programming language in comparison to C++. That should allow users of the latter to rapidly get acquainted with the notions of the former. Thank you for forwarding any comments you may have on to author, at the address that appears in the document.

From: Jeff Creem <jcreem@yahoo.com>
Date: Tue, 28 Dec 2004 13:36:54
Subject: Re: newbie - OOP in Ada Set and Get Methods
Newsgroups: comp.lang.ada

You probably need to step back and read a few of the Ada tutorials rather than trying to hack-and-whack C++ in Ada. [...] this tutorial is short and is a very good starting point: http://www.adahome.com/Ammo/cpp2ada.html [...]

Articles about Memory Management in Ada

From: David Botton <david@botton.com>
Date: Sat, 27 Nov 2004 19:53:46 -0500
Subject: Re: Memory management in games
Newsgroups: comp.lang.ada

Please see these two articles in the advanced section of the AdaPower Source Code Treasury

Memory Management with Storage Pools (Anh Vo)

http://www.adapower.com/index.php?Command=Class&ClassID=Advanced&CID=222

Ada Answers - The Ada Lecture Series

URL: http://www.adacore.com/aa_lectures.php

The Ada Lecture Series

Learn more about Ada through this informative series of university lectures and conference presentations given by some of the foremost experts on the language.

“Course: Ada Past, Present and Future”

Robert B. K. Dewar is a professor at NYU, President of AdaCore, and one of the early figures in the development of Ada. In this lecture given at the Massachusetts Institute of Technology, Mr. Dewar gives an overview of the history of Ada, which includes the motivation for its conception, the story of its development, and the role Ada plays in present day programming. The lecture also covers the fundamental ideas behind Ada, its influence on other languages, and the truths and myths associated with the language.

Ada-related Tools

Simple components 1.8

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 26 Dec 2004 20:35:30 +0100
Subject: ANN: Simple components v 1.8
Newsgroups: comp.lang.ada

The version 1.8 is here:
http://www.dmitry-kazakov.de/ada/components.htm

It finally introduces an abstract persistent storage interface. Storage interface objects are accessed through handles an so can be referred by persistent objects which then may act as proxies to the persistent storage rather than just being saved and restored. (That is of course possible too)

A framework for concrete persistent storage interfaces is provided for the data bases capable to identify objects using keys.

This version also provides ready-to-use implementations of the interface:

1. An ODBC persistent storage interface based on GNADE.

2. An APQ persistent storage interface based on APQ 2.1 by Warren W. Gay VE3WWG. (Though our disagreements with Warren, I have decided to support it along with ODBC)

Examples of use are supplied.

[See also same topic in AUJ 25-4 (Dec 2004), p.185. -- su]

PragmARC - PragmAda Reusable Components

From: PragmAda Software Engineering <pragmada@earthlink.net>
Date: Mon, 27 Dec 2004 04:59:14
Subject: New Release of the PragmAda Reusable Components
Newsgroups: comp.lang.ada

A new release of the PragmAda Reusable Components (PragmARCs) is now available. This release primarily adds some additional sorting algorithms.

The PragmARCs are available from: http://home.earthlink.net/~jrcarter010/pragmarc.htm

Errors, suggestions, and comments are welcome at pragmada@earthlink.net.

[See also same topic in AUJ 25-2 (Jun 2004), p.48. -- su]

ASIS for GNAT: GCC-3.4.4

From: Martin Krischik <martin@krischik.com>
Date: Sun, 28 Nov 2004 19:47:41 +0100
Subject: [Announce] ASIS for GNAT gcc-3.4.4-20041123 released
Newsgroups: comp.lang.ada

Hello,

It has been some time since I last prepared an ASIS release. Nothing new, just compiled for a current gcc-3.4.4.

See http://gnat-asis.sourceforge.net/

[See also "ASIS for GNAT: New Project and First Versions" in AUJ 25-2 (Jun 2004), p.56 -- su]

AVR-Ada

From: Rolf Ebert <rolf.ebert@gmx.net>
Date: 25 Nov 2004 13:02:06 -0800
Subject: [Announce] AVR-Ada V0.2.1 released
Newsgroups: comp.lang.ada,comp.arch.embedded

We are proud to announce a new release of AVR-Ada, one of the first GCC based Ada compilers targeting 8-bit microcontrollers.

You can get the project description and some documentation at: avr-ada.sourceforge.net

The SF development pages with the download section are at: www.sourceforge.net/projects/avr-ada

AVR-Ada is available in source form only. Binary packages of the cross compiler hosted on Linux and Windows are expected to appear with future releases of cdk4avr (cdk4avr.sourceforge.net) and WinAVR (winavr.sourceforge.net).

Feel free to join the mailing list at: http://lists.sourceforge.net/mailman/listinfo/avr-ada-devel.

It has quite low traffic.

Please use SF's bug reporting system for guiding future development of AVR-Ada.

The aim of the AVR-Ada project is to make the gcc based Ada compiler GNAT available for the AVR microcontrollers.

More specifically the project provides:

- a working compiler based on the existing AVR and Ada support in gcc (V 3.4.3)

- a minimalist Ada runtime system

- an AVR specific support library containing all the necessary part descriptions as Ada package specs.

The current distribution of AVR-Ada is V0.2.1. It is based on gcc-3.4.3. The Ada compiler of gcc-3.4 is considerably better than gcc-3.3. In the AVR-Ada project I had never problems with the Ada compiler itself. It is very stable.

The Ada run time system (RTS) on the other hand is not even a *run* time system. It is more a compile time system :-). All files in the RTS are only needed at compile time. As a consequence we don't have support for exceptions nor for tasking (multithreading).

There is some AVR specific support. Type and interface definitions, timing routines, eeprom access, UART, and most importantly the necessary definitions for most AVR parts.

Sample programs in the apps/ directory show how to use the compiler and the library. This includes the tutorial program from the avr-libc distribution translated to Ada.

The documentation consists of the pages at avr-ada.sourceforge.net. A copy of the pages is in the directory AVR-Ada/web/ for offline reading.

We modified the compiler patches to fit cleanly to gcc-3.4.3. They probably also fit in previous gcc-3.4.x releases; I never tried.

The "freestanding" patch is now considerably shorter since part of what it does (avoid some code in the binder file) is now provided directly in gcc. The corresponding binder options were previously called "standalone", but that expression is used elsewhere in GNAT.

You can now use "Library Projects" with AVR-Ada. I.e. you can have gpr files with "Library_Name", etc. The only permitted value for "Library_Kind" is "static". This feature is used for the Avr library.

gcc-3.4.3 now fully supports -fdata-sections (PR14064) and -ffunction-sections for AVR targets in C and Ada. This is particularly useful for embedded systems where code and static data must not be wasted.

[See also "GNAT Compiler for AVR Targets" in AUJ 25-1 (Mar 2004), pp.7-8. -- su]

Ada 2005 Numerics Library Implementation

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: Wed, 02 Feb 2005 22:36:46 +0100
Subject: Re: Calculs et tracés numériques en Ada
Newsgroups: fr.comp.lang.ada

[Translated from French – su]

> I am looking for the way to perform rapid numerical calculations (hence, with no interpreted language). I had immediately thought of C++ and at a dedicated library … I have also considered Fortran, but I have been wondering whether Ada might respond to my needs.
> In fact, I would simply like to manipulate matrices and vectors as well as perform additions, subtractions and multiplications, scalar products, vector products, member-to-members multiplications (in the way of Scilab). That library should also handle all problems with incoherent dimensions. Overall, I am looking for something rather trustable.
> Do you possibly have any suggestions?

There exits an ISO norm (ISO/IEC 13813:1996(E)) which defines a library for matrix manipulation with Ada. That norm should in principle be integrated with Ada 2005. While waiting for that, you may find an implementation of it at: http://dflwww.ece.drexel.edu/research/ada/ [...]

[See also "Ada0Y.Directories - AI-248 Implementation" in AUJ 24-3 (Sep 2003), p.138. -- su]

GNAVI Progress

From: David Botton <david@botton.com>
Date: Fri, 19 Nov 2004 07:14:52 -0500
Subject: Not Announcement of GNAVI IDE
Newsgroups: comp.lang.ada

I wouldn't call this an announcement since this isn't a release.... but GNAVI now has a functional IDE that lets you:

Create new projects

Edit projects

Create new windows

Add windows already created to the project

Delete windows from the project

Edit the body, spec and XML of the window

You can compile and run from the IDE

Using the XML specs, you describe your GUI and automatically the source of the body and spec is modified (along with other parts silently) with the changes.

The editor is lame and things need much work, but progress is happening! The outline and GUI layout views are already in the works.

http://www.gnavi.org

[See also "A community Windows binding" in AUJ 25-4 (Dec 2004), pp.188-188, and "Delphi and GNAVI" and "GUI Programming for Beginners" in this issue -- su]

From: David Botton <david@botton.com>
Subject: GNAVI Updates
Date: Fri, 26 Nov 2004 11:28:07 PST
Newsgroup: comp.lang.ada

Figured I would let people know about GNAVI progress:

Ok, I've got a fully working Outline view for GUI building (ok, it still needs a more work, but it is functioning). I am already using it here and there now to do some of the work on the GNAVI IDE as it is.

Here is today's screen shot:
http://www.gnavi.org/images/snap1.jpg

It's ugly, but it works :-) It will all get cleaned and polished in time.

I need to put in all the controls now in to the datastore/controls.xml, set up the properties and handlers to show even the ones not set yet and a bit more. I am itching to get on to the GUI layout view very soon.

I don't have time to package a snapshot today, but if any one wants to play, as always it's in CVS.

Just check out modules: gnatcom, gwindows and gnavi (see http://www.gnavi.org for information) [...]

From: David Botton <david@botton.com>
Date: Thu, 23 Dec 2004 00:34:52 -0500
Subject: GNAVI Progress
Newsgroups: comp.lang.ada

Just wanted to give people a heads up on progress on the GNAVI project, The Open Source Visual RAD answer to Delphi/Kylix and the Visual Dark Side

The IDE is in full swing and can now even do some basic Visual layouts of controls, etc.

You can see a snap shot of the current work on the IDE at:
http://www.gnavi.org/images/snap1.jpg

There is a Win32 snapshot (for the brave) available at
http://www.gnavi.org/gnavi/gnavi.zip

For more information see:
http://www.gnavi.org

XIA - XPath In Ada

From: Marc A. Criley <mc@mckae.com>
Date: Fri, 31 Dec 2004 20:48:55
Subject: Announce: XIA 0.60 now available
Newsgroups: comp.lang.ada

Version 0.60 of XIA (XPath In Ada) is now available on the McKae Technologies website at www.mckae.com/xia.html.

This version of XIA is a beta release that completes the initial implementation of the XPath 1.0 specification. Therefore this release has sufficient capabilities implemented that one may now consider it for actual XML applications. Improvements and fixes to known and to-be-discovered bugs will be incorporated into subsequent releases.

There are two significant bugs and a known limitation: The parser treats "and", "or", "not", "div", the axis names ("child", "ancestor", etc.), and node type names ("text", etc.) as reserved words. Meaning that if an XML document uses such a term as an element tag, referencing it as part of the path in an XPath expression will generate a syntax error. Predicate expressions that specify the union, via '|', of two node-sets will also cause a syntax error. Lastly, the id() core library function is implemented, but inoperable, as XMLAda does not yet implement the Get_Element_By_ID function.

The txia_test.txt file, containing a list of XPath queries that seriously exercise the predicate filtering capabilities of XIA, has been updated to 140 queries, and the distribution now also includes an "expected results" file for running the txia test sequence.

As this is a beta release, reports of errors (either in operation or in the nodes retrieved) would be appreciated. Please provide the XML document (or readable fragment), the query that was submitted, and a description of what was expected.

[See also same topic in AUJ 25-4 (Dec 2004), p.188. -- su]

Bindings for OpenAL (Open Audio Library)

From: Aurele <aurele.vitali@gmail.com>
Date: 22 Jan 2005 17:30:20 -0800
Subject: OpenALada
Newsgroups: comp.lang.ada

OpenAL (for Open Audio Library) is a software Application Program Interface (API) to a computer's audio hardware. OpenALada is a new Ada binding to OpenAL.

[OpenALada is distributed under GNU LGPL -- su]

Check it out here: www.OpenALada.com

From: Aurele <aurele.vitali@gmail.com>
Date: 4 Mar 2005 06:56:35 -0800
Subject: OpenALada
Newsgroups: comp.lang.ada

OpenALada and OpenALdemo v1.1 have been updated and tested with ObjectAda v7.2.2 and GNAT v3.15p.
www.OpenALada.com

Database Source Name Parser

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Fri, 14 Jan 2005 07:35:25 +0100
Subject: [ANN] Data Source Name parser (ODBC etc.)
Newsgroups: comp.lang.ada

Some time ago when discussing APQ or Ada and databases, Brian May suggested URL-like strings describing database connections. A parser library for such strings (data source names) is now available at:
http://home.arcor.de/bauhaus/Ada/dsn.html

The library is at version 0b.2, that is, it has bugs. Some are known, some are fun. The distribution contains a small sample program that allows interactive checking of DSNs.

Documentation is available in source and as:
http://home.arcor.de/bauhaus/Ada/dsn.pdf

I have tried to make the interface simple, any comments as to whether it is usable or whether it should go down the drain will affect following versions.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Mon, 31 Jan 2005 21:24:53 +0100
Subject: [ANN] data source names (ODBC etc) parser, v 0b.3
Newsgroups: comp.lang.ada

The second release of the data source name parser library is now available at http://home.arcor.de/bauhaus/Ada/dsn.html.

The most notable change is a further simplified interface. My plan is to leave the interface the way it is now, unless comments and critique say that it should change.

The library is stabilizing, but nevertheless the version number, which is 0b.3, announces beta software.

A small interactive demo program, contained in the distribution, shows what this is all about, for correct, and for incorrect input:

```
DSNT>
db2://georg@tcp+localhost:4321/parts
Yes.
-  system name: db2
-         user: georg
-     password:
-     protocol: tcp
-       server: localhost
-         port: 4321
-     database: parts
-   parameters:

DSNT>
db3://georg@tco+localhost:4321/parts
 1: database system name not known,
skipping "db3"
 13: not a known protocol, skipping "tco"
 1: input is not valid, but could be
corrected
Maybe.
```

The program is a simple demonstration of how to use the library subprograms in an application.

Ada+SQL

From: Stephane Richard <stephane.richard@verizon.net>
Date: Thu, 04 Nov 2004 13:20:36
Subject: Re: Linux mag de novembre
Newsgroups: fr.comp.lang.ada

[Translated from French – su] Chris wrote:

> I am an Ada novice (I have already ordered the book “Programming in Ada”.)
> I am looking for code examples or for a “cook-book” site, especially about: sqlite / prog objet / gtkada

In addition, another very compact SQL system may interest you: http://www.readyideas.com/ada+sql1.htm

It’s a method that seems capable of integrating SQL with the Ada language.

[See also "PGSQL - PostgreSQL Minimal Binding" in AUJ 25-4 (Dec 2004), p.190. -- su]

Public Release of GNAT Programming System IDE (GPS)

From: Arnaud Charlet <charlet@province.act-europe.fr>
Date: Wed, 22 Dec 2004 14:34:34 +0100
Subject: Announce: GNAT Programming System 2.1.0
Newsgroups: comp.lang.ada

A Christmas present ;-)

Enjoy!

Ada Core is pleased to announce the release of GPS 2.1.0 Academic Edition, the GNAT Programming System IDE, including binaries for the GNU/Linux, Solaris and Windows platforms.

Designed by programmers for programmers, the GPS IDE integrates the GNAT Ada 95 tools within a single visual development environment. GPS is Free Software. This version is intended for use in academic and Free Software projects.

GPS is available at http://libre.adacore.com/gps

New features include, among other things:

- Integrated python interpreter

- More powerful customization capabilities

- Better multi-language support (in particular C and C++)

- Improved efficiency

- Generic Version Control support

From: Arnaud Charlet <charlet@gnat.com>
Date: Mon, 3 Jan 2005 10:26:14 +0100
Subject: Re: Announce: GNAT Programming System 2.1.0
Newsgroups: comp.lang.ada

> Is there any changelog for changes since the 2.0.1 GAP bundle edition? I installed it two days ago

You will find this information along with GAP 1.1 which also includes GPS 2.1.0 and was released around the same time.

[See also "AdaCore - GPS 2.1.0" in AUJ 25-4 (Dec 2004), pp.193-194 and "ACT - Public Release of GNAT Programming System IDE (GPS)" in AUJ 24-3 (Sep 2003), p.146. -- su]

AdaDoc 2.1 - Documentation tool for Ada Package Specifications

From: Vincent Decorges <vega01@sf.net>
Date: Tue, 28 Dec 2004 09:00:52
Subject: ANN: AdaDoc 2.1
Newsgroups: comp.lang.ada

We are announcing the release of AdaDoc version 2.1.

This version includes new modules to generate Texinfo and StrongHelp documentations.

Thanks to Bent Bracke and Stéphane Rivière for their contributions.

AdaDoc is a tool for developers using the Ada95 programming language. Its goal is to create documentation in different format from a specification package.

The program analyzes the specification (by controlling its syntax), then the draft to write a document with the desired format. The format of exit depends on the selected module. AdaDoc writes a file XML (temporary) containing all information necessary to the modules to write the other formats (HTML, Latex, etc...). The writing of a module for other formats is very easy.

AdaDoc is freely available to the following address: http://adadoc.sf.net

One can find there:

- binary versions for Linux and Windows.

- a user guide (English and French).

- a guide of creation of module (English and French).

- sources of the software (English).

- the complete documentation of the project (only in French).

- a mailing list to be informed of the new release.

AdaDoc is under the GPL.

[See also "AdaDoc 2.01 - Documentation tool for Ada Package Specifications" in AUJ 24-3 (Sep 2003), pp.145-146 -- su]

Ada-related Products

AdaCore - GNAT Pro Nominated for 'Product of the Year'

Date: Wed, 26 Jan 2005
Subject: GNAT Pro Nominated 'Product of the Year' in Jupiter Media's Datamation Awards

New York, USA - January 26, 2005 - AdaCore is proud to announce that its flagship product, GNAT Pro has been nominated for 'Product of the Year' in Jupiter Media's Datamation Awards. Our distinguished co-nominees are Mozilla's Firefox, VMWare's Workstation and Novell's SUSE Linux Enterprise Server. The Datamation Product of the Year 2005 Awards recognize Datamation readers' choices for achievement and innovation in enterprise software and hardware products.

To learn more about the awards (or better yet, vote for GNAT Pro) [go to http://awards.jupitermedia.com/index.php/42311/ -- su]

AdaCore - GNAT Pro 5.03a

From: Jamie Ayre <ayre@adacore.com>
To: Ada User Journal
Date: Mon, February 21, 2005 4:19 pm
Subject: [Ada User Journal] AdaCore articles for consideration

GNAT Pro 5.03a - new platforms, new tools and new Ada 2005 support

This latest version of the GNAT Pro toolsuite offers close to 100 new features including support for new platforms and targets, the introduction of the gnatmetric and gprmake tools, and support for many of the new Ada 2005 features.

The GNAT Pro list of supported platforms and targets continues to grow

with the addition of ppc-darwin for Mac OS users, ERC32 targets for the space market, and Pentium targets for VxWorks, among others.

Integrated into the GNAT Pro toolset, gnatmetric calculates a set of commonly used industry metrics, which allows you to estimate the size and better understand the structure of your source code base, and also to satisfy the requirements of certain software development frameworks. Gprmake tool provides gnatmake-like multi-language building capabilities based on the GNAT project files.

As part of the ongoing standardization activities for Ada, the language is reviewed periodically to see if corrections or new features are warranted. AdaCore is directly involved with the Ada 2005 language amendment process, and has been steadily implementing the approved new Ada 2005 features. For full details of the new features already implemented in GNAT, please read our paper "GNAT and Ada 2005" available on our web page dedicated to Ada 2005 (http://www.adacore.com/ada_2005.php)

GNAT Pro 5.03a also offers greater efficiency and stability for its users:

- Zero Cost Exceptions support is now available on many more platforms including certain cross configurations.

- The 5.03a back end, derived from gcc 3.4, offers a high degree of maturity and stability.

GNAT Pro's availability over an ever increasing number of native and cross platforms, combined with exceptional, responsive support from Ada experts provides for focused, time saving, project development.

[See also "ACT - GNAT Pro 5.02a" in AUJ 25-2 (June 2004), pp.58-59. -- su]

AdaCore - GNAT Pro Toolsuite for ERC32

From: Jamie Ayre <ayre@adacore.com>
To: Ada User Journal
Date: Mon, February 21, 2005 4:19 pm
Subject: [Ada User Journal] AdaCore articles for consideration

New GNAT Pro Toolsuite for ERC32

GNAT Pro for ERC32, a flexible cross-compilation system supporting the Ravenscar tasking profile on top of a bare ERC32 computer, is now available. It is designed for mission-critical real-time space applications, especially those that have to meet safety standards.

Developed under ESA (European Space Agency) sponsorship, AdaCore targeted the compiler to the ESA's standard processor for spacecraft on-board computer systems, the ERC32, which is a radiation-tolerant SPARC V7 processor. Available host platforms are x86 Linux and SPARC Solaris.

The static and simple tasking model defined by the Ravenscar profile allows a streamlined implementation of the Ada run-time library directly on top of bare computers. Its reduced complexity, together with its configurability, makes it an excellent choice for mission-critical space applications in which certification or small size is needed.

The developer can choose from several predefined run-time libraries, each corresponding to a particular set of run-time Ada features, or, even more flexibly, configure a tailored library reflecting exactly the set of features that are used.

Also as part of the ESA contract, AdaCore has developed a comprehensive test suite that checks compliance with the Ravenscar profile and correct behavior of specialized features (such as the last-chance exception handler mechanism) and supplemental tools (such as the debugger).

IPL (www.ipl.com) were also involved in the development providing their AdaTEST 95 tool targeting the ERC32 compiler.

"We are very pleased with this development which more than ever opens the space market to Ada, and to the use of a state-of-the-art software development environment for mission-critical applications." - Dr. José Ruiz, AdaCore lead engineer tasked with completing the port.

AdaCore - GNAT Pro Available for Mac OS X

From: Jamie Ayre <ayre@adacore.com>
To: Ada User Journal
Date: Mon, February 21, 2005 4:19 pm
Subject: [Ada User Journal] AdaCore articles for consideration

GNAT Pro Compiler and Debugger Available for Mac OS

As part of the 5.03a release, AdaCore is pleased to announce the availability of the GNAT Pro toolsuite for the Mac OS X and Mac OS X Server platforms, Apple's award-winning UNIX-based operating system.

GNAT Pro for Mac OS X/PowerPC, brings together the exciting features of the Ada programming language with the innovative G5 processor, and makes a perfect base for developing robust applications from high-performance routines to large-scale servers.

GNAT Pro is based on the same long-standing compiler technology (GCC) as Apple uses, ensuring it benefits from the latest state-of-the-art developments, and making it the best Ada solution on Mac OS X.

GNAT Pro is already available on more platforms that any other Ada compiler, and in porting to Mac OS X, AdaCore hopes to attract a whole new segment of the developer communities to the Ada programming language.

About AdaCore

AdaCore, a privately held company founded in 1994 with major offices in New York City and Paris, produces and provides expert support for the GNAT Pro family of open-source Ada 95 and multi-language software development environments. The GNAT Pro toolset is used in a wide range of industries including aerospace, defense, energy, transportation, media, banking, communications, automobile, and medical software.

About GNAT Pro

Based on the GNU GCC technology, GNAT Pro is available on more platforms than any other Ada compiler and is the only implementation of the complete Ada 95 language, including all the Specialized Needs Annexes. GNAT Pro, the professional edition of the GNAT technology and the premier Ada and multi-language development environment on the market, is used on enterprise-critical projects encompassing areas such as low-level communications control, high-integrity real-time applications and large-scale distributed systems.

Aivosto - Visustin v3 flowcharts Ada code

From: Aivosto Oy <vbshop@aivosto.com>
To: Ada User Journal
Date: Wed, 9 Mar 2005 12:01:22 +0200
Subject: PR: Visustin v3 flowcharts Ada code

For Immediate Release

Date: 9 March 2005

Contact: Tuomas Salste (vbshop@aivosto.com)

Aivosto has updated Visustin, a flowcharting tool. Visustin v3 reverse engineers Ada code to diagrams. Recent improvements include automated flowcharting of an entire system, robust charting of large modules, generation of Visio flow diagrams and support for ASP, JSP, PHP and Fortran code.

Visustin produces flow charts from complex, unstructured source code in 18 programming languages. A detailed chart visualizes all the code with comments attached. A bird's eye view shows the core logic leaving out unnecessary details.

Automated drawing of Microsoft Visio flow diagrams is possible with Visustin. Freeing the user from manual drawing, Visustin can document existing systems in a short time. The charts are useful as technical documentation and for learning how an algorithm works. Visustin

supports multi-page printing and saves the charts in all standard image and web formats.

Visustin flowcharts the following languages: VB, VB.NET, VBA, ASP, QuickBASIC, C/C++, C#, Java, JSP, JavaScript, COBOL, Fortran, Pascal/Delphi, Perl, PHP, T-SQL, PL/SQL and Ada.

A functional evaluation copy is available for free download at www.aivosto.com.

Pricing: US$299/Standard Edition, US$449/Pro Edition.

Supported operating systems: Microsoft Windows 95/98/ME/NT/2000/XP/2003

Optionally supports: Microsoft Visio 2002/2003

Tuomas Salste, Aivosto Oy

Kylanvanhimmantie 16, 00640 Helsinki, Finland

vbshop@aivosto.com, www.aivosto.com

DDC-I - SCORE Challenges Legacy DACS Product

Date: Fri, 29 Oct 2004
Subject: DDC-I's Versatile SCORE® IDE Challenges Legacy DACS Product
URL: http://www.ddci.com/news_SCORE_challenges_DACS_release.shtml

DDC-I's Versatile SCORE® IDE Challenges Legacy DACS Product

Venerable "small, tight code" generating DACS goes head-to-head with SCORE in a large customer application - with surprising results

Phoenix, AZ - October 29, 2004 - Responding to a customer query as to the code size performance of the powerful SCORE® (Safety Critical, Object-oriented, Real-time Embedded) integrated development environment (IDE) compared to their legacy DACS toolset - with a strong reputation for producing tight code - DDC-I engineers proudly reported back a not-so- surprising discovery: a virtual dead heat.

"This project proves SCORE® is capable of generating target executable code size on par with DACS, while also facilitating mixed use and debugging of multiple high level languages targeting several key embedded processors," explains DDC-I Engineering Manager and SCORE® Product Champion David Mosley. "We also identified several improvements that will soon allow SCORE® to produce consistently smaller code than DACS."

Created as the logical successor to the widely used DACS, SCORE® presents powerful leverage to real-time embedded system developers, mixing application development among different languages, including Fortran, C, Embedded C++ and Ada 83/95, while realizing significant cost and time savings during inevitable transitions to new processor technologies.

The first multi-language, multi-target, multi-host IDE for real-time safety-critical embedded system developers based on open standards, the latest beta build of SCORE® (2.5) compiled and linked the customer code alongside DACS 4.7.14 for their current 80x86 protected mode target. Code size down to the target was the benchmark, for a complex application - including tasking, storage management and exceptions - consisting of 539 files and roughly 261,130 lines of Ada95 source code. Minor changes were necessary for DACS compilation in Ada83.

The initial results: DACS 1,196,966 bytes, SCORE® 1,231,306 bytes. Just 2.8 percent apart, attributable mainly to more complex Ada95 tasking and exception management requirements. As SCORE® is also designed to take advantage of the extended features and increased memory in the latest generation of hardware, simply "de-tuning" its capabilities reduced the gap to 1.7 percent. Additional changes are calculated to bring SCORE® as much as 4.7% below DACS, with even greater code size savings when auto-inlining is suppressed. Optimizing access to outer scope variables and a number of other modifications will likely further improve code size reduction.

Including a highly reliable compiler, seamlessly integrated multi-language debugger and two small, exceptionally fast tasking & non-tasking run-time systems, SCORE® offers developers with a wealth of legacy code a mature means to migrate to the latest technology, as well as extending existing source with newer embedded code. Based on Win32 and OSF/Motif, the Windows- oriented "point-and-click" character of the SCORE® GUI incorporates project tools, online help, tool activation and other efficient features -- with the command-line option always available for byte-conscious power users.

DDC-I - SCORE Debuts Fortran Compiler

Date: Mon, 29 Nov 2004
Subject: SCORE® Version 2.5 Debuts Fortran Compiler
URL: http://www.ddci.com/news_SCORE_version_2_5_debuts_fortran_release.shtml

SCORE® Version 2.5 Debuts Fortran Compiler

New Fortran 77 compiler continues the ever-expanding range of DDC-I's multi-language, multi-target migration options for legacy developers

Phoenix, AZ - November 29, 2004 - In keeping with the ever-increasing importance of legacy code migration among embedded system developers, DDC-I today announced the addition of a native Fortran compiler in version 2.5 of the maturing SCORE® (Safety Critical, Object-oriented, Real-time Embedded) integrated development environment (IDE), in addition to several key component updates.

"Several customer-driven improvements to SCORE® are included in SCORE® 2.5," explains David Mosley, DDC-I Engineering Manager and SCORE® Product Champion, "but number one is direct compilation and debugging of Fortran, which dramatically decreases the complexity of legacy code migration by allowing developers to maintain code in the original Fortran and enabling programmers to move easily between Ada 95, C, Embedded C++, and Fortran source."

According to Mosley, the new Fortran compiler, based on the ANSI X3.9-1978 Fortran(77) standard, supports all current processors, as well as the MIL-STD-1750A, added to version 2.5. Full support for the popular Dy4-181 PowerPC board and multi-language debugger support for the powerful Abatron JTAG probe are also new.

Recently performance tested head-to-head against its predecessor, DDC-I's mature DACS compiler, at the request of a customer, version 2.5 of the SCORE® compiler generated final code size results on par with DACS. Improvements found during the test process are already being integrated, beginning with support for inlining of non-local programs. Constant recognition has been greatly improved, especially when dealing with complex structured constants, resulting in significantly smaller code and data. Machine code insertions are now context-sensitive.

Based on Win32 and OSF/Motif, the Windows-oriented "point-and-click" character of the SCORE® GUI incorporates project tools, online help, tool activation and other efficient features.

Today, COTS solutions regularly meet project requirements at a fraction of the cost of in-house development, while integrated suites like SCORE® make the next big leap, facilitating flexible migration from different languages and platforms into a uniform future. The process improvements that modern tools and languages make possible reach directly to the bottom line, where thousands of lines of reused code - now including Fortran - can reduce costs and increase programmer productivity. Outdated tools migrated to SCORE® gain multi-language, multi-target capability while placing minimal restrictions on future development.

For customers evaluating SCORE®, DDC-I also offers their popular Migration

Assessment Packages, offering on-site needs assessment, evaluation and a comprehensive report describing the complexity and functionality of software migration including: current systems, utilization, capacity and scalability, resource and skills planning, education and training, cost evaluation, risk assessment and any additional recommendations. MAPs are individually shaped and priced, to help customers with complex applications achieve project goals on-time and budget.

DDC-I - SCORE Integrates ARINC 653 RTOS

Date: Wed, 01 Dec 2004
Subject: SCORE® Integrates VxWorks' ARINC 653 RTOS
URL: http://www.ddci.com/news_SCORE_VxWorks_ARINC_653_release.shtml

SCORE® Integrates VxWorks' ARINC 653 RTOS

DDC-I increases flexibility for SCORE developers using Wind River's robust, partitioned VxWorks AE653 RTOS

Phoenix, AZ - December 01, 2004 - Always working to help safety-critical embedded system software developers control costs and compress time to market, DDC-I today announced integration of the versatile SCORE® (Safety Critical, Object-oriented, Real-time Embedded) integrated development environment (IDE) with the Wind River VxWorks® AE653 RTOS, offering complete ARINC 653-1 compliance and DO-178B Level A certification.

"Statistics show the code load of a typical embedded system doubling about every two years and an average of sixty-six percent of projects over budget, while a third fall short functionally," explains DDC-I Engineering Manager and SCORE® Product Champion David Mosley. "Meeting software development goals is getting harder all the time, and we continue to increase the capabilities of SCORE® specifically to help developers keep beating the odds."

According to Mosley, the aerospace and defense industry demands a standardized OS with robust partitioning, which allows uncertified applications to co-exist with fully certified applications. An ARINC 653 compliant OS, such as VxWorks AE653, meets this need.

Already chosen for development, operation and maintenance of the systems driving the fuel boom ACU on the new 767 Global Tanker Transport Aircraft, integration of the VxWorks product increases the functional reach of SCORE for developers already using the AE653 RTOS.

Including a highly reliable compiler, seamlessly integrated multi-language debugger and the integrated AE653 RTOS, SCORE® offers developers with valuable legacy code, especially in Ada and Fortran, a mature means to migrate to the latest targets and technology, as well as the ability to extend existing source with newer embedded code.

"Designed specifically for the development of high-integrity embedded systems, SCORE® provides a unified ARINC 653 solution for the world's highest performance aerospace applications, while also offering a flexible, integrated turnkey solution for every application where safety and reliability are number one," Mosley concludes.

DDC-I - Product Development Calendar for 2005

Date: Mon, 20 Dec 2004
Subject: DDC-I Product Development Calendar for 2005
URL: http://www.ddci.com/news_product_development_calendar_2005.shtml

DDC-I Product Development Calendar for 2005

Phoenix, AZ - December 20, 2004 - In today's fast-moving technology environment, embedded system developers designing and maintaining safety-critical systems face long-term maintenance commitments that demand uncompromising tool support from vendors. With over two decades of experience, DDC-I has long recognized the value of maintaining an adaptable tool development model capable of changing as customer needs change.

"We see our work as a shared commitment with our customers, counting on their fresh input and ideas to keep our products growing and adapting, as well as pursuing our own development initiatives based on trends and industry developments," explains David Mosley, DDC-I Engineering Manager.

According to Mosley, definitely on the development calendar for 2005 are support for Wind River's VxWorks and the addition of VxWorks AE653 to the SCORE® (Safety Critical, Object-oriented, Real-time Embedded) integrated development environment (IDE). As SCORE's Product Champion, he confirms the 2005 completion of current projects to add the 1750A target and Fortran support to the already well-developed IDE, alongside current multi-language support for C, Embedded C++, and Ada 95.

Also slated are scheduled updates to mature DACS and TADS development tools still flying high throughout the safety-critical real-time embedded system industry, reflecting DDC-I's standing commitment to their customers.

Just over the horizon, a JOVIAL rehosting to the Windows environment looms likely, as well as the migration of the SCORE® IDE to Eclipse and the addition of support for Venturcom (www.vci.com) RTX and Phar Lap to SCORE®. Out beyond radar range, Java and Linux support -- as well as i960 and 68xxx target additions -- are under evaluation, with a weather eye on support for the upcoming Ada 05 standard.

"With ten-year maintenance and support agreements typical on the back-end of most embedded systems development programs," Mosley concludes, "developers need software tools that can adapt alongside the rapid technology changes that necessarily drive their work - and DDC-I remains committed to building them."

DDC-I - Riding High with Cassini-Huygens

Date: Tue, 01 Feb 2005
Subject: 2.2 Billion Miles and Counting: Riding High with Cassini-Huygens
URL: http://www.ddci.com/news_riding_high_with_casini_huygens_update.shtml

2.2 Billion Miles and Counting: Riding High with Cassini-Huygens

Working with three major project contractors on Cassini-Huygens, software coded with DDC-I tools is successfully orbiting Saturn

February 1, 2005 -- Phoenix, AZ -- Burning the twilight Cape Canaveral sky, a Titan IV-B/Centaur launch vehicle lifted the Cassini-Huygens spacecraft toward the stars on October 15, 1997, covering the first miles of a 2.2-billion-mile-long "slingshot" ride through the solar system toward Saturn. Too large a mass to shoot straight at the ringed planet, four separate gravity-assisted "turns" hurled the spacecraft along its interplanetary trajectory, passing Venus (twice), the Earth and Jupiter before approaching Saturn in mid-2004.

Cassini-Huygens is the most complex interplanetary spacecraft ever built; it represents the best technical efforts of the United States and 17 European nations involved in the mission. Onboard the dual-mission craft, consisting of the JPL-built Cassini orbiter and an ESA-built Huygens probe destined for the surface of Saturn's largest moon Titan, is embedded systems software coded by American and European engineers using DDC-I software developments tools.

Space missions define the outer performance envelope of well-used industry terms like "mission-critical," and Cassini's systems have performed flawlessly since lift-off. Cassini and the Galileo spacecraft were used in concert to study Jupiter between October 2000 and

March 2001, taking advantage of their dual vantage points to observe the shape of the magnetosphere and the effects of solar wind. On July 1, 2004, the main engine fired and Cassini-Huygens was captured by Saturn. Angling through a gap between Saturn's F and G rings, the craft made its closest arc around the planet to begin the first planned orbit of a four-year primary mission.

Bolted to Cassini in "sleep mode" -- and awakened once every six months for a three-hour instrument and engineering checkup -- the Huygens probe was released before reaching Titan's atmosphere, on December 24, 2004. Umbilical cut, Huygens spun gently away from Cassini on a ballistic trajectory toward Titan. Two days later, Cassini adjusted course to miss Titan and setup optimal signal reception for telemetry streaming back during the probe's descent.

The Cassini-Huygens mission will answer fundamental questions about the evolution of planets through extensive study of Saturn, its rings, magnetosphere, Titan and other icy moons. The Saturn system represents a laboratory -- the equivalent of a miniature solar system -- where scientists can seek answers to fundamental questions about the physics, chemistry and evolution of planets and the conditions that give rise to life. Saturn may contain much of the primordial cloud's gases not trapped by the Sun. The largest moon Titan is thought to harbor organic compounds important in the chain of chemistry that led to life on Earth. Too cold to support life now, it is a "frozen vault" that may show what the early Earth was like.

On January 14, 2005, Huygens entered Titan's atmosphere at 13,500 miles per hour. Designed to withstand the extreme cold of space (about -330F) and the intense heat of atmospheric entry (over 2,700F), the probe used atmospheric drag to reduce speed until a series of parachutes began deploying at 870 miles per hour. During the descent, instruments sampled the physical properties at different levels in the atmopshere and captured the first images of the moon's surface.

An exotic world with geophysical processes similar to Earth operating under alien conditions, many of Earth's familiar forms occur on Titan, but the chemistry involved is quite different. Instead of liquid water, Titan has liquid methane. Rather than silicate rocks or dirt, Titan has hydrocarbon particles settling out of the atmosphere. On Titan volcanoes spew ice. Huygens touched down in liquid methane mud -- and quickly took more samples. Thirty minutes later, Cassini's antenna would be out of range.

Cassini's planned tour of the Saturn system includes 52 close encounters with seven of Saturn's 31 known moons. Gravity-assist flybys of Titan, as close as 590 miles, will permit high-resolution radar mapping of Titan's surface to produce vivid topographic maps, as well as providing "slingshot" propulsion. Each orbital path is a mission: the imaging of Titan, fly-bys of selected icy moons, occultations in Saturn's rings and crossings of the ring plane. Fly-bys will be made of other major moons and Saturn's Polar Regions and equatorial zone. The prime mission officially concludes on June 30, 2008, four years after Saturn arrival and 33 days after the final Titan flyby on May 28 aims Cassini for a follow-on Titan flyby one month later, ready to proceed with additional missions if resources allow. A few remarkable scientific discoveries and several billion miles from now, DDC-I will still be flying high among the rings of Saturn.

[See also "Cassini-Huygens Reachs Titan" -- su]

PegaSoft - BUSH AdaScript Business Shell

From: Ken O Burtch <kburtch@sympatico.ca>
Date: Sat, 15 Jan 2005 18:50:00 -0500
Subject: ANN: BUSH 1.0.1
Newsgroups: comp.lang.ada

BUSH (Business Shell) combines the capabilities of a Unix shell, PHP, GCC and PostgreSQL into a uniform design for rapidly designing secure, reliable Web templates. Using AdaScript, based on GNAT, BUSH promotes code reuse: scripts and templates can be compiled with GNAT/GCC or ported to JVM with JGNAT or .Net with A# with minor changes. It can also work as a replacement for a Bourne shell with native SQL support, and is a general scripting language. BUSH is part of ABEE, a proposed Ada system-wide development environment.

BUSH 1.0.1 was released January 15, 2005 for Linux or FreeBSD running PostgreSQL. Recent changes include a FreeBSD port, improved documentation with new tutorials and a new debug mode for web templates.

BUSH is available for download from http://www.pegasoft.ca/bush.html.

Ports to other operating systems welcome.

[See also same topic in AUJ 25-1 (Mar 2004), p.11. -- su]

Praxis HIS - SPARK Team honoured by ACM SIGAda

From: Rod Chapman <rod.chapman@praxis-cs.co.uk>
Date: 11 Nov 2004 07:44:51 -0800
Subject: ANN: SPARK User Group meeting - Final Programme
Newsgroups: comp.lang.ada

I'm pleased to say that the final programme for the 2004 SPARK User Group meeting is now available on www.sparkada.com

If you're interested in attending (and we haven't already invited you!) then please drop us a line at sparkinfo@praxis-his.com

Date: November 2004
Subject: SPARK Team honoured by ACM SIGAda
URL: http://www.praxis-his.com/sparkada/events.asp

We're pleased to announce that SPARK Team has been awarded the 2004 ACM SIGAda award for outstanding contribution to the Ada community. At the SPARK User Group meeting, the award was dedicated to Professor Bernard Carré - the founder of Program Validation Limited and principal designer of SPARK.

Praxis HIS - SPARK Release 7.2

From: Rod Chapman <rod.chapman@praxis-cs.co.uk>
Date: 10 Jan 2005 09:30:54 -0800
Subject: ANN: SPARK Release 7.2
Newsgroups: comp.lang.ada

We're pleased to announce the immediate availability of Release 7.2 of the SPARK Language and Toolset.

This release incorporates several significant improvements. Full details are available in the Release Note, which is available from www.sparkada.com.

For readers of the SPARK Textbook, upgrade packages are also available from www.sparkada.com including the new language definition, manuals and demonstration tools for IA32/Windows and IA32/Linux. These upgrade packages are also available from the "SPARK Book" page of www.sparkada.com as usual.

Supported professional customers are being sent upgrades now. Academic users and tool-partners will receive their upgrades shortly.

Some technical highlights of this release include:

Language:

Full-range record subtypes are now supported.

Rules for passing array elements as "in out" parameters have been relaxed (this significantly eases the construction of iterator algorithms.)

String constants that are constrained by their initializing expression are allowed.

Finally, instantiation and use of Unchecked_Conversion is permitted.

Examiner:

Flow analyser more accurately models "for" loops that have a static range.

New VC Generator model of "for" loops correctly models all loops, including those with a dynamic range where variables controlling the loop exit are modified in the loop body.

Declaration of subprograms in the private part of a package is implemented.

VC Generator produces hypotheses showing that local variables are "in" their designated subtype.

A new "brief" error message mode eases integration with EMACS and GPS by producing "gcc style" errors that can be recognized by such environments.

Information flow errors are produced by default in a new easier-to-read format.

Refinement proofs between the private and full view of a private type are now supported.

The VC Generator can now produce replacement rules for composite constants, under the control of a new command-line switch and a new annotation.

Simplifier:

New tactics for quantified expressions, structured object updates and scalar inequalities. These give a significant improvement in Simplifier "hit rate" (aka Completeness) for common SPARK idioms, and particularly for proofs of exception freedom.

SPARKSimp supports multi-processor machines for improved throughout (NOT available with the Demo toolset...)

Other:

A new utility "SPARKMake" that automates the production of Examiner index files and meta-files.

[See also "Praxis Critical Systems - SPARK Book Upgrade Packages Available" in AUJ 25-3 p.125. -- su]

References to Publications

Article about GTK+ and Ada at Linux Magazine

From: Lionel Draghi <Lionel.DRAGHI@fr.thalesgroup.com>
Date: Thu, 4 Nov 2004 12:37:27 +0100
Subject: Linux mag de novembre
Newsgroups: fr.comp.lang.ada

[Translated from French – su] Folks, let me this opportunity to advertise the November issue of the GNU Linux magazine in which Simon Descarpentries has published a very favourable article on Ada and GtkAda.

Perhaps that may be of use and I hope it will stem many other such articles!

[The article is titled "Gtk+ et Ada: Le duo gagnant?". See the contents of the GNU Linux Magazine France issue 66 at the URL http://www.linuxmag-france.org/produit.php?produit=372 -- su]

DDC-I Online News

[Extracts from the table of contents. See elsewhere in this News section for selected items. -- su]

From: jc <jcus@ddci.com>
To: 17D November 2004 Online News US <jcus@ddci.com>
Date: Mon, 1 Nov 2004 14:33:35
Organization: DDC-I
Subject: Real-time Industry Updates - News from DDC-I

DDC-I Online News, Real-Time Industry Updates - November 2004, Volume 5, Number 11 - [http://www.ddci.com/news_vol5num11.shtml] A monthly news update dedicated to DDC-I customers & registered subscribers.

This Month:

DDC-I Versatile SCORE IDE Challenges Legacy DACS Product - Impressive results prove SCORE can compete with the best of the best!

Controlling Teams and Projects in SCORE - White paper details how SCORE's project structure allows good team development

Thoughts from Thorkil - Tasking and Priority Inversion in Ada

The Power of Gratitude - A small investment with the possibility of huge returns.

From: jc <jcus@ddci.com>
To: 20D December 2004 Online News US <jcus@ddci.com>
Date: Wed, 1 Dec 2004 16:58:49 -0700 (MST)
Organization: DDC-I
Subject: Real-Time Industry Updates - News from DDC-I

DDC-I Online News, Real-Time Industry Updates - December 2004, Volume 5, Number 12 - [http://www.ddci.com/news_vol5num12.shtml] A monthly news update dedicated to DDC-I customers & registered subscribers.

This Month:

SCORE(R) Integrates VxWorks' ARINC 653 RTOS - A flexible, integrated, turnkey solution for safety critical applications

DDC-I Releases SCORE(R) V.2.5 - Debuts Fortran 77 compiler and other updates

Tech Talk - Using "Force_Reset_On_Quit" in the SCORE(R) Multi-Language Debugger

Why Can't We All Play Nice - Ideas For Team Development: Recognize Your Differences and Focus on Shared Concerns

From: jc <jcdk@ddci.com>
To: 21D January 2005 Online News US <jcdk@ddci.com>
Date: Fri, 31 Dec 2004 12:34:07 -0700 (MST) (20:34 CET)
Subject: Real-Time Industry Updates - News from DDC-I

DDC-I Online News, Real-Time Industry Updates - January 2005, Volume 6, Number 1 - [http://www.ddci.com/news_vol6num1.shtml] A monthly news update dedicated to DDC-I customers & registered subscribers.

This Month:

2004 Product Sales Above Forecast - Customer Care Focus Widens Customer Base

DDC-I Product Development Calendar for 2005

Thoughts from Thorkil - Floating Point Concepts and the 80x86 Implementation

Touching the Void - Be Prepared to Break the Rules

AdaCore to Present Several Papers at Ada Europe 2005

Date: Wed, 26 Jan 2005
Subject: AdaCore to Present Several Papers at Ada Europe 2005
URL: http://www.adacore.com/pressroom_13.php

Paris, France and New York, USA - January 26, 2005 - AdaCore is pleased to announce that it will be presenting the following papers and tutorials at Ada Europe 2005 which takes place this June in York, England:

Ada 2005 Abstract Interfaces in GNAT, Gary Dismukes and Javier Miranda

The Application of Compile-Time Reflection to Software Fault Tolerance using Ada 95, Andy Wellings and Pat Rogers

GNAT Pro for On-Board Mission-Critical Space Applications, José Ruiz

A Comparison of the Mutual Exclusion Features in Ada and the Real-Time Specification for Java, Ben Brosgol

ERB: The ESA Ravenscar Benchmark, Roman Berrendonner, Jerome Guitton, Nicholas Roche

Real-Time Java for Ada Programmers (Tutorial), Ben Brosgol

Software Fault Tolerance (Tutorial), Pat Rogers

More about Ada 2005 at: http://www.adacore.com/ada_2005.php -- su]

Ada and Java

JGNAT Portable Ada Code

From: Marc A. Criley <mc@mckae.com>
Date: Sun, 12 Dec 2004 13:37:57
Subject: Re: JGNAT
Newsgroups: comp.lang.ada

Andrew Carroll wrote:

> Can a program be created such that JGNAT can be used as well as compile without JGNAT? Like be able to output both Java classes and regular "executables" given one "set" of code?

Yes. But IF, and ONLY IF, you stick to pure, vanilla Ada code. That means no Java libraries for user interfaces or anything. (You could of course always create wrapper packages that are then implemented either natively or with Java libraries.)

And JGNAT 1.1 is pretty hoary by now; it had bugs, and was almost production-ready, but it was dropped as an AdaCore maintained product some time ago. And if I recall correctly (it's been over 3 years since I last used it), while it worked pretty good with JDK 1.2, tasking broke with JDK 1.3, and I vaguely recall reports that it just pretty much didn't work much at all with 1.4.

Unless you want to pick up the JGNAT sources and bring them up to snuff (which would be great! :-), I'd be hard-pressed to think of any practical use for it--unless you want to play around with it on old JDKs.

For something along the same lines, you might want to look at A#, which is Ada for the .NET platform. It is being actively maintained, albeit as a university computer science research project: http://www.usafa.af.mil/dfcs/bios/mcc_html/a_sharp.html

Converting Java To Ada?

From: Brian May <bam@snoopy.apana.org.au>
Date: Wed, 15 Dec 2004 13:48:35 +1100
Subject: Re: Converting Java To Ada?
Newsgroups: comp.lang.ada

Conrad wrote:

> Hi, are there any tools for converting Java source to Ada source (i.e. adb and ads)?

Yes.

It is called a "programmer". These tools walk around on two legs and eat pizza. Make sure you get the correct model, you want a model that supports Ada and Java. Also make sure you have plenty of pizza.

Sorry, couldn't resist ;-).

From: Nick Roberts <nick.roberts@acm.org>
Date: Wed, 15 Dec 2004 04:43:35 +0000
Subject: Re: Converting Java To Ada?
Newsgroups: comp.lang.ada

In fact, I would concur with this answer.

If you only need to convert from Java to Ada because a Java executive is not available for the environment in which you wish to run the program, then an automated conversion tool might make sense. But this seems like an unlikely scenario, somehow.

If, on the other hand, you need to convert a Java program into Ada, and then subsequently maintain the program (in Ada), then I suggest you use the aforementioned bipedal pizzivorous tool.

There are good reasons for this. Ada provides many language constructs that have no direct analogy in Java (and there are a few Java constructs whose Ada analogues are ugly as hell). Probably, only a human -- and a very skilled one, at that -- can make the sophisticated transformations required to convert the Java idioms into appropriate Ada ones.

Oh, and I've heard that these humans drink copious quantities of coke, as well as eating pizza at all times of the day and night (plus the occasional Alka-Seltzer).

From: Tommy Zhu <tommy@cnluton.com>
Date: Fri, 17 Dec 2004 13:19:32 +0000
Subject: Re: Converting Java To Ada?
Newsgroups: comp.lang.ada

I would suggest to add this topic to Ada Faq: The Ada Funnies.

Ada Inside

Cassini-Huygens Reaches Titan

From: Vinzent 'Gadget' Hoefler <nntp-2005-01@t-domaingrabbing.de>
Date: Wed, 19 Jan 2005 15:27:12 +0000
Subject: Have you missed this great success story?
Newsgroups: comp.lang.ada

Hey, what's up with you guys? Almost a week ago we landed on Titan and nobody here cares?

<URI:http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=33006&fbodylongid=1099>:

> The software is based on a top-down hierarchical and modular approach using the Hierarchical Object-Oriented Design (HOOD) method and, except for some specific low level modules, is coded in Ada. The software consists, as much as possible, of a collation for synchronous processes timed by a hardware reference clock (eight Hz repetition rate). In order to avoid unpredictable behaviour, interrupt-driven activities are minimised. Such a design also provides for better software observability and reliability.

I guess, a few links on Ada-Success stories could be updated...

[See also "DDC-I - Riding High with Cassini-Huygens" in this issue -- su]

About the Ada Job Market

From: Jeff C r e e.m <jcreem@yahoo.com>
Date: Tue, 14 Dec 2004 02:51:11
Subject: Re: Ada job market?
Newsgroups: comp.lang.ada

Mike wrote:

> Just a general question, how is the market for ada programmers? Are veterans programmers able to move over to Ada?

The answer to both questions is that it depends where you are. The job sites are not the best gauge since some companies don't include it as a requirement in the listing (they assume no one has experience)....

As near as I can tell half the places doing embedded work put XP/Visual C++ in their ads just because they think that is what people want to see.

From: Luke A. Guest <laguest@abyss2.demon.co.uk>
Date: Tue, 14 Dec 2004 14:07:07 +0000
Subject: Re: Ada job market?
Newsgroups: comp.lang.ada

Why put it there then? Surely they're not doing their jobs properly by putting that? But it's not like you can find jobs anywhere else because all programming jobs are through agencies, FFS. Agencies are a joke!

From: Martin Dowie <martin.dowie@baesystems.com>
Date: Tue, 14 Dec 2004 14:50:26 -0000
Subject: Re: Ada job market?
Organization: BAE SYSTEMS
Newsgroups: comp.lang.ada

Luke A. Guest wrote:

> Well, I've seen one which is for a *Junior* programmer, they list the Ada83 as a requirement as well as having security clearance AND experience of Rational Apex. That makes me think they're not looking for a junior at all. A junior (to me) would be someone with no experience, who can get SC and can learn Apex/Ada83 (who uses 83 anyway?)

Lots of projects!!!

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Tue, 14 Dec 2004 17:03:53 +0000 UTC
Subject: Re: Ada job market?
Newsgroups: comp.lang.ada

> What was the point of Ada 95 then? ;-)

Lot of _new_ projects!!! ;-)

From: Martin Krischik <martin@krischik.com>
Date: Tue, 14 Dec 2004 09:23:03 +0100

Subject: Re: Ada job market?
Newsgroups: comp.lang.ada

I have seen lot's of offerings in the UK (http://www.theitjobboard.com). While I don't mind moving to the UK, and don't mind working in defence - 90% of them need security clearance. Well I am german so my chances are pretty slim here.

From: Martin Dowie <martin.dowie@baesystems.com>
Date: Tue, 14 Dec 2004 09:59:39 -0000
Organization: BAE SYSTEMS
Subject: Re: Ada job market?
Newsgroups: comp.lang.ada

Not at all! You might be barred from some jobs but I don't see much of a problem working on loads of things, e.g. Eurofighter Typhoon.

Indirect Information on Ada Usage

[Extracts from and translations of job-ads and other postings illustrating Ada usage around the world. -- su]

Date: Tue, 16 Nov 2004 15:00:00
Subject: Ada Programmer near Frederick MD
URL: http://www.adaic.com/jobs/

Experienced Ada programmer needed for federal office in suburban Maryland (near Frederick). This is a permanent, full-time position working on enhancement and development to existing systems. Excellent salary, benefits and growth opportunities for right candidate. Prefer 4+ years background in Ada development but will consider talented beginner. DoD Secret clearance is needed for position - will consider "clearable" experienced candidates only.

Date: November 30, 2004 at 15:00:00
Subject: Ada Software Engineers - Charleston SC area
URL: http://www.adaic.com/jobs/

We have a client located in the Charleston, SC area that is beginning to ramp up for an 18 month project. Need 6-10 Software Engineers Junior through Senior. Will be working with Ada and C++. Experience with embedded systems in a realtime avionics environment a plus. U.S. Citizenship required. We will begin deploying resources in January. If you would be interested, please send your resume and required W2 bill rate. Unless incorporated and carrying appropriate insurances.

Date: Wed, 5 Jan 2005 15:00:00
Subject: Experienced Ada Programmer - Ft. Detrick MD
URL: http://www.adaic.com/jobs/

[...] seeking an experienced Ada programmer to work as part of a team maintaining and upgrading computer-based communications system in a military environment.

The selected candidate will have five or more years of experience in software development (Ada) and experience with UNIX and Oracle. A DOD Secret clearance (and the ability to obtain a Top Secret clearance) is required.

This is an excellent opportunity to join a well-established team in a nurturing environment. We offer competitive salaries as well as an excellent benefits package.

Date: Mon, 10 Jan 2005 16:00:00
Subject: Senior Level Ada Developer - Virginia
URL: http://www.adaic.com/jobs/

Job Description: Real-Time Embedded Ada software Engineer, VxWorks, Rational Apex as part of an established team and established project. Clearance not necessary.

Date: Thu, 30 Dec 2004 15:00:00
Subject: Ada 95 Software Engineer - San Diego CA area
URL: http://www.adaic.com/jobs/

Wanted: Mid-Level Ada programmer.

San Diego location.

3-6 mos. contract to direct. Definitely will convert the right person. May consider direct placement. Not interested in a pure contractor. Clearance is nice, but clearability is vital. Benefits start on day 1 of direct hire and are very good (my assessment.)

Job Description:

The NIDS II Project provides the targeting capability for the U.S. Strategic Command. The team is in need of a skilled Ada/Ada95 software engineer for a fun, new development effort involving target/aimpoint optimization. Development will be using the GNAT/GLAD tools in a distributed Unix environment.

Education: Bachelor's degree in Computer Science or a scientific discipline.

Required Skills: 5+ years experience in software engineering several years focus with Ada or Ada95.

Desired Skills: Knowledge of J2EE, CMMI and Unix a plus. A Secret clearance is also desired.

Schedule and Status: Full-time (1st shift)

Able to Obtain Clearance: Top Secret

Possess Clearance: N/A

Relocation: Local or regional candidates are preferred, as relocation is not approved for this position.

Date: Sat, 18 Jan 2005 15:00:00
Subject: Embedded Software Engineer - San Diego CA area
URL: http://www.adaic.com/jobs/

[...] technical staffing firm that assists companies and organizations with information technology staffing needs in every industry. Our client has an excellent Embedded Software Engineer opportunity available for the right candidate. If you are looking for an exciting, innovative opportunity with the chance to excel, then this is the opportunity for you.

Summary:

Temp to perm

Ada Development

Real-time Embedded Software

Object Oriented

C++, UML

Secret clearance recently active

Date: Thu, 23 Dec 2004
Subject: UNIX Software Developer (Brussels, Belgium)
URL: http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/jobs

Contract for one year in Brussels - starting ASAP.

Participate in the development, enhancement and testing of software systems used for co-ordination in Air Traffic Management (ATM) within Air Traffic Flow Management (ATFM) and between ATFM, Air Traffic Control, Air Space Management, Airports and Airlines.

The task includes analysis of requirements, implementation constraints, performance and reliability aspects. Production of detailed technical specifications, including test specifications and participation in the implementation of the ATFM applications written mainly in Ada95.

1. Detailed Design (specification of parts of the system)

2. Implementation (design, code & test)

3. Participate in test automation preparation and implementation

4. Writing of documentation.

5. Participation in on-call service.

Essential: Engineer with at least 2 years of experience in design and implementation of Application Software using OO techniques. Mainly Ada. C++ knowledge is an advantage, Ada 95 knowledge is an advantage.

Date: Mon, 10 Jan 2005
Subject: Ada / C++ Analyst Developer (Vlaams-Brabant, Belgium)
URL: http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/jobs

[...] The service lines offered by [us, an IT Service and Consulting company,] are:

* Information Systems Development based on open platforms, techniques and methods (J2EE, UML, SOAP,

Application Servers, .NET, Relational Databases, etc.)

* Application & Infrastructure Support (system management [Unix or Windows], network management, application maintenance, etc.)

* Implementation of Document, Content and Knowledge Management Systems (documentary databases, classification tools, conversion tools, portal building technologies, XML/SGML, Business Intelligence tools, etc.)

* Real-time Systems Development (specialized languages, real time kernel, etc.)

* Implementation of Quality Assurance Processes (Quality Assurance & Control)

* Audits and Technical Architecture Studies

We are current looking for an Ada / C++ Developer

* Several years of experience in a similar function using Ada 95 / Linux.

* Experienced in Air Traffic Management (ATM) is a major advantage.

* Full university level, or graduate in Computer Science or engineering.

* Fluent in French, Dutch and English.

* Position Type: Full Time, Permanent.

Date: Mon, 24 Jan 2005
Subject: Ada 95 Software Developer (Belgium)
URL: http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/jobs

My client has an urgent requirement for a Software Developer in Ada 95 on Linux. You will have 3-5 years Ada 95 development experience within Air Traffic (ATC) environment. Knowledge of working on Air traffic control systems or large mission critical systems is desirable.

Your role will include:

* Detailed Design (specification of parts of the system)

* Implementation (design, code & test)

* Participate in test automation preparation and implementation

* Writing of documentation.

* Excellent opportunity for a long term contract.

Statistics about Ada Adoption in Industry and University

From: Adrian Hoe <byhoe@greenlime.com>
Date: 30 Oct 2004 05:04:09 -0700
Subject: Need statistic for Ada
Newsgroups: comp.lang.ada

I am planning to write a paper about how fast Ada has been adopted at work and education over the last 5-10 years. Is there any statistical data about the adoption of Ada in the industries and universities?

Could someone provide me some pointer?

Ada in Context

Ada Suitability as a Game Development Language

From: John B. Matthews <jmatthews@wright.edu>
Date: Mon, 22 Nov 2004 03:00:10 GMT
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

Jeff Houck wrote:

> I'm new to Ada but have 15+ years background in C/C++ and x86 assembly. I skipped Pascal and Java (yuk)and have used the usual scripting languages, Perl, Python, Tcl/Tk, etc... I'm bored with coding C/C++ and would like to try something new... 8^) Anyway, I'm really intrigued by Ada and I'd like to "go out on a limb" and see how Ada measures up to C/C++ for game development. I'm thinking about both the 2D and 3D regimes, such as an isometric style game in 2D and a FPS for 3D.
> You don't see Ada mentioned anywhere (that I know of) in game development circles. Is there a specific reason why? I'd like to hear from any in this News Group with an opinion or insight on the subject. Thanks!

I've enjoyed tinkering with linxtris: <http://sourceforge.net/projects/linxtris>

From: David Botton <david@botton.com>
Subject: Re: Ada suitability as a game dev language
Date: Mon, 22 Nov 2004 18:10:06 -0500
Newsgroups: comp.lang.ada

You can find a list of known freely available games with Ada source code at: http://www.adapower.com/index.php?Command=Class&ClassID=Applications

From: Stephane Richard <stephane.richard@verizon.net>
Date: Mon, 22 Nov 2004 03:23:44 GMT
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

If you go to my website (http://www.adaworld.com) in the Ada Projects then Binding Projects page....you'll see a project of interest there...AdaOpenGL

Likewise, in the library projects you'll see one called Engine_3D this one is 100% Ada...no dependencies on other libraries, and you'll see it's got a lot to offer.

From: Martin Krischik <martin@krischik.com>
Date: Mon, 22 Nov 2004 09:15:28 +0100
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

If you don't mind 20% more typing and 90% less debugging than you are right here ;-) .

Ada is true multi purpose language and that includes games. Interesting for games might be:

* the ability to define you own float types (speed vs. precision).

* arbitrary sized integer (12 bit integer - no prob).

* packed or un-packed arrays and records (speed vs. size).

* true multi dimensional arrays.

* choice OO or non-OO programming.

[...]

From: Nick Roberts <nick.roberts@acm.org>
Date: Tue, 23 Nov 2004 21:39:17 -0000
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

The feature of Ada that Martin seems to have egregiously omitted is its especial support for multi-tasking, something that can be of especial importance for some kinds of games programming. Often it's actually quite fun to write a game in Ada where you simply have a task for each automaton (or indeed every entity which has individual behaviour). There is at least one MUD written in Ada for precisely this reason.

DoD (and other NATO) contractors often use Ada to create (most of) the software for military and commercial flight simulators and related systems. This is often because: they write real flight systems in Ada, so they already have Ada programmers and expertise; much of those flight systems can be directly re-used in the simulators anyway; Ada is great at multi-tasking. If you consider a flight simulator (or battlefield management simulator, etc.) to be a kind of grown-up's big game (and I know the people who 'play' them do!), then you could consider Ada to be a language much used for games, in fact.

>> You don't see Ada mentioned anywhere (that I know of) in game development circles. Is there a specific reason why?

> Because they don't know better. It takes a month or two to see how cool Ada truly is.

I suspect the answer is more accurately "For all the same reasons that most programming shops don't use Ada." Of course, ignorance is undoubtedly one of those reasons, but (as we have discussed

in this news group recently) the main reason seems to be the lack of a big commercial player willing to support and promote the Ada language (or an implementation of it).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 22 Nov 2004 09:36:02 +0100
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

I believe that Ada might be very promising for game development. What comes in mind first is:

1. Ada has much more carefully designed numeric model than C++. 3D graphics requires a lot of non-trivial computations.

2. Ada is highly portable. Games are usually developed for many platforms.

3. Ada has integrated tasking. Real-time strategy, simulation games etc.

4. Ada requires much less debugging. Games are large and complex software products with a very short testing phase.

From: Dale Stanbrough <dstanbro@bigpoop.net.au>
Date: Mon, 22 Nov 2004 09:31:54 GMT
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

The major platforms would be (in no particular order)...

Windows

XBox (windows again)

Playstation

There is no Ada compiler for the Playstation, so your argument has less force than you think.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 22 Nov 2004 12:02:16 +0100
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

Probably you mean commercial games, for you do not mention Linux and Apple. But in that case, I believe it would be no problem for a big software player to persuade ACT, Aonix or RR software to target Playstation. Or other platforms of even greater interest like mobile phones, for instance.

From: Luke A. Guest <laguest@abyss2.demon.co.uk>
Date: Tue, 23 Nov 2004 08:20:50 +0000
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

David Botton wrote:

> Is there any GCC compiler for Playstation? If so, than a port would not be that problematic.

Yes, Sony provides it on their PS2 Pro site with source. SN Systems' compiler is based on GCC and from what I can tell, so is Metrowerk's PS2 compiler. These don't come with source, but SN did provide the home-brew people with their version of the GCC compiler.

From: David Botton <david@botton.com>
Subject: Re: Ada suitablity as a game dev language
Date: Mon, 22 Nov 2004 18:14:07 -0500
Newsgroups: comp.lang.ada

Alex R. Mosteo wrote:

> In Windows you may get away with GNATCOM, but that's just for Windows.

I have an example of doing Direct X programming using Ada with GNAVI at:

http://www.gnavi.org/index.php?Command=Class&ClassID=GWindowsWin32

From: Luke A. Guest <laguest@abyss2.demon.co.uk>
Date: Mon, 22 Nov 2004 16:04:07 +0000
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

Well, like the others have said, Ada is a multipurpose programming language, much like C and C++ are. Anything that can be done in C or C++ can also be done in Ada.

The problems you'll find are bindings. These you *can* do yourself with a little bit of work, it's not that difficult to do, just a bit confusing in certain areas, like data types and getting something more Ada like (a thick binding compared to a thin binding), i.e. there is the Interfaces.C package which gives you basic C types like int, etc. You would then later prefer thick bindings which use the native Ada types or your own (preferable). This can produce a lot of code due to all of the conversions necessary to get the data into the correct data types.

Now concerning types, In C/C++ you normally define your own types which are platform independent, e.g. a 32-bit signed int, a 16-bit unsigned int. Now a question that was raised in another thread concerned me about this, a lot of people suggested that you should let the compiler decide what type to use for a value, which is not what you want in games, you really need to have something that is portable, especially for data formats, i.e. I have a field which specifies how many vertices I have in a model, I've decided that that needs to be a 32-bit type, therefore over all platforms it needs to be a 32-bit type. Now you can use representation clauses to specify the size, but surely you need to specify the type also?

Anyway, any C libs can be imported no problems into Ada, C++ is not defined and probably won't be.

As for platforms, GCC exists for practically all platforms, yes even PS2, Sony provide the source to it if you're a registered developer. According to Robert Dewar, the GCC-3.4.x MIPS code generation needs work, so porting it to PS2 could be tricky. Also, as Sony have had the brilliant idea of combining all of the GNU source (binutils, gcc, gdb, newlib, etc.) all into one directory, and they've modified the build scripts, it's not possible to extract diffs for different packages (easily). It's also not possible to drop in GNAT-3.3 (I think that's the latest version they support) into the gcc/gcc/ directory and expect an Ada compiler to pop out, it won't, I've tried it.

Another problem will be porting the runtime, there's practically no documentation on it, so that'll be difficult.

For a game, you really need to conserve space, so another thing you'll want to do is to stop the compiler from generating elaboration code for a lot of stuff (if not all of it). GNAT can be told not to produce any and everything seems to work ok, yes I've even tried array's with strange boundaries. You'll also need to do this if you decide to use a very basic Ada compiler without a runtime, also doable - especially on a console. But it'd be better with one.

Another problem that games programmers hear is "don't use exceptions; it'll slow down the game!!" Well, this might have been true a long time ago, but GNAT does provide a new exception mechanism "zero-cost" which provides tables rather than setjmp()/longjmp() calls. I haven't tried it.

But for desktop platforms, it's a piece of piss really. OpenGL bindings do exist, I've been modifying them - haven't touched them for a while, but they do work. They're a combination of two sets of bindings, although a generator really needs to be written to take the opengl.spec files and convert them to packages: http://www.abyss2.demon.co.uk/projects/ada/index.html

Believe me, there's nothing I'd rather do than do develop in Ada, but for games, it's just not viable...yet. Until other companies start to realise that programming massive games in C/C++ is really a waste of time (and a lot of time at that, because they have such stupid schedules) it's just not going to happen.

My latest project (a game engine) is being written in C++ basically, because I will be able to sell it if it's written in a language that other developers can use straight away. If it's written in Ada, I won't be able to sell it.

> You don't see Ada mentioned anywhere (that I know of) in game development circles. Is there a specific reason why?

And you won't, a lot of games programmers haven't been to University, and if they have, they didn't do computer science, or they did but it's before Ada

was being hyped (~1990's), so they wouldn't have heard of it. They still have the mentality of "everyone else uses C/C++, so it must be great."

I thought about writing an article for Gamasutra, but it ended up being a bit of a rant (like this ;-)), so I didn't finish it. Maybe I will when I leave this company ☺

From: Dani <danielcheagle@ig.com.br>
Date: Mon, 22 Nov 2004 22:28:21 -0200
Subject: Re: Ada suitablity as a game dev language
Newsgroups: comp.lang.ada

My primary intent when I learnt Ada was making games. I've not created a game yet but it is my "heart's dream game" working in progress, in my spare time.

The Key points (initially) in my choice was: (my game is a massive on-line rpg; server part is in Linux and the client part is in Linux and windows, but can be "easily" portable to others platforms if the tools used exist on these other platforms (normally "true"))

* "tasks" (concurrency directly in the language) the compiler takes care for us of OS dependency part. The source code is really very more portable. pthreads, native threads, win tasks,etc ? no... just Ada tasks. The program takes advantage automatically if exists more that one processor/cpu, e.g. a server side game (to support the load charge and the responsiveness necessary). switch from time-slicing only to true real-time and time-slicing without

* speedy: because the Ada compiler known much more aspects of the program (see *.ali parts for one example) it can easily make a program in Ada 2,5 more rapid that the same program write in C. (This is controlled with compiler switches.)

Other characteristics that made me very very happy were

* Making a program, thinking 'in Ada' is much more pleasure and much less fatiguing.

* Reading a program written in Ada (e.g. to maintain a program or to understand what other programmer wrote) is more understandable and much less stressful. Ada privileged the reader.

* The compiler caught much more mistakes in the code, making the final program much more bugs free.

The libs and bindings I intend to use are:

*adasockets for the network part of the engine.

*adaopengl combined with gtkada-gl-area for the part of opengl. (Note: gtkada-gl-area in MS-Windows environment is more or less trick to compile (the lib). but just for now this is not a problem.)

*GtkAda for the GUI and other administrative parts.

The tools that I recommend are:

*Glade-2 for the initial GtkAda design and looking the source code generated is a good teaching for the guys that are not genius guru in gtk (my case :-) (Please do not confuse glade-2 (GUI generation) with the glade (gnat part of annexe E distributed Systems))

*Gnat-GPS to deal with source code.

And

* Gnat compiler (or gcc 3.4.x with Ada support)

Well..., I hope that helped you.

</flame on> please no flames :-) <\flame off>

It's of seven roughs, Dani.

From: David Botton <david@botton.com>
Date: Mon, 22 Nov 2004 18:08:43 -0500
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

Take a look at video 8. The folks at SGI talk about why Ada was superb for use in exactly the sort of task you are talking about.
http://www.adapower.com/index.php?Command=Class&ClassID=AdvocacyVideos

From: Jeff Houck <jhouck@northrim.net>
Date: 24 Nov 2004 02:05:17 +0100
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

Wow! Thanks for all the insight, information and useful URLs regarding this post! I can see that interacting with this newsgroup is going to be very interesting... 8^)

I'm going to give this some serious thought. In reading through everyones comments and looking at the examples, I'm convinced that Ada can, and perhaps more importantly should, make a showing in the game dev realm. Ada has simply got so much going for it, how can it continue to be "left out"?

But, before I get myself in too deep, I'm going to work out a general specifications list for the development of a simple Ada/OpenGL/GLFW based engine. Initially I expect it will do "2D in 3D" (textured polys as sprites) and if that is successful, a full 3D environment. The OpenGL/GLFW part is entirely based on David Holms AdaOpenGL work. I recently contacted David inquiring about his project and while he hasn't worked on it for a little over a year, it's quite complete. He would like to receive any bug reports and screenshots showing the library in action. Thanks David!

Please keep in mind that I'm looking to produce a "proof of concept" at this stage. This will be my first attempt at coding a game engine so I'm sure it'll be a learning experience. 8^)

I'll post again later with some ideas to bounce off anyone who cares to reply or possibly participate.

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: Wed, 24 Nov 2004 20:26:38 +0100
Subject: Re: Ada suitability as a game dev language
Newsgroups: comp.lang.ada

[...] There is a project on SourceForge called the Generic GNU Game Core: http://sourceforge.net/projects/g3c/

"G3C provides the main features for 3D-game developers: 3D rendering engine based on openGL, collision detection, physical rules, p2p network... A game-sample will be available, binding a war-game, a flight simulator, a first person shooter, a MMOG..." [...]

You will probably find that Ada, as a language, has everything you need for game development. Now, you need to evaluate the surroundings of the language and see if they meet your requirements:

- availability of an Ada compiler for your target platforms

- availability of libraries or bindings

- ease of distribution of your work. Ada programs require an Ada runtime library, plus any other libraries you used.

If you are a hobbyist using and targeting free software, you are in luck. The Ada compiler is available (GNAT) and included in most GNU/Linux distributions and in FreeBSD. There are several free libraries available (AdaSDL, AdaOpenGL, Generic GNU Game Core, GtkAda). The distribution maintainers will package any libraries you require, making distribution easy.

If you plan to write free software, you may also want to look for other developers to join you. This might be difficult because few game developers are willing to learn Ada. OTOH, with Ada you would need less time and fewer people to develop a game than you would in C.

From: Nick Roberts <nick.roberts@acm.org>
Date: Thu, 13 Jan 2005 17:51:59 +0000
Subject: Re: Use of Annex E in a game server (Linux/GNAT)
Newsgroups: comp.lang.ada

Luke A. Guest wrote:

> I've been talking to a friend who has written an online game and said he needs a server, I offered to help out. So, I've been thinking about it and not only would Ada be a great choice for a normal server, I was thinking about the distributed annex, and using that to balance the load over to more servers as they get added/needed.
> Can anyone advise on whether this is a

> good thing to do? What to avoid (if anything), tip, tricks, etc?

I can't give much personal advice, but there is an Ada MUD project: http://members.aol.com/drveg/mud/sam.html

Maybe something there could be handy.

I guess you would have a tagged (abstract limited) type to represent a server, and then use remote access types to direct calls to servers. Could be cool.

Ada Seen by Fortran Programmers

From: Israel t <rambam@bigpond.net.au>
Subject: Obsolete like Fortran ? You wish !
Date: Sun, 09 Jan 2005 07:37:33
Newsgroups: comp.lang.ada

Brian May wrote:

> I think many people treat Ada as an obsolete language, much like Fortran (for example)

Fortran is very much in use. It is also extensively used for new code, especially in the HPC community.

Fortran 95 and F (a clean subset of Fortran) continue to attract hundreds of millions of dollars in new development.

The Fortran 2003 specification draft has been finalised and is available at http://www.kcl.ac.uk/kis/support/cit//fortran/john_reid_new_2003.pdf

From: James Van Buskirk <not_valid@comcast.net>
Date: Mon, 31 Jan 2005 17:56:40 -0700
Subject: Re: Shortcut logicals (was: Re: F200x)
Newsgroups: comp.lang.fortran,comp.lang.ada

James Giles wrote:

> Well, I have the F2003 FCD open right now, in §7.4.3 Masked array assignment - WHERE. So, perhaps you can point out what part of it resolves the issue I raised:
> cond1 .and. cond2 .andthen. cond3 .and. cond4
> Is COND4 permitted to be evaluated or not? I don't see that WHERE tells me. It has a concept of "pending control" which I see can be applied on nested constructs, but in the above expression, not of the terms actually present are part of any "pending" condition. Or, are you talking about some other aspect of WHERE entirely?

OK, I have added comp.lang.ada, so hopefully someone there can answer the question about the effect of precedence on your snippet, and also discuss the ergonomics of Ada short- circuiting logical operators, or whatever they call them.

[...]

From: Dan Nagle <dannagle@verizon.net>
Date: Tue, 01 Feb 2005 17:55:09 GMT
Subject: Re: Shortcut logicals (was: Re: F200x)
Newsgroups: comp.lang.ada,comp.lang.fortran

Hello,

Martin Krischik wrote:

> This is of course a cross post with comp.lang.fortran and I wonder how they see our solution to the problem. Is it suitable for Fortran 2003 as well or do they need another solution?

This thread started in c.l.f discussing J3's efforts to add andthen and orelse operators to Fortran. J3 was unable to form a consensus regarding the operator precedence, and tried a different tack.

The way the Fortran standard is written makes it fairly difficult to delay evaluation of arguments to operators or to functions.

J3 often examines the way Ada does something when seeking ideas. Personally, I believe Ada is a very well designed language, and I'm much more comfortable getting ideas from Ada than from many other languages.

The fact that others disagree is probably why this thread now appears beyond c.l.f.

From: James Giles <jamesgiles@worldnet.att.net>
Date: Tue, 01 Feb 2005 18:25:23
Subject: Re: Shortcut logicals (was: Re: F200x)
Newsgroups: comp.lang.ada,comp.lang.fortran

Dan Nagle wrote:

> J3 often examines the way Ada does something when seeking ideas. Personally, I believe Ada is a very well designed language, and I'm much more comfortable getting ideas from Ada than from many other languages.

Ada has a lot of very useful ideas. I was against cross-posting this thread to the Ada group because I had assumed that Ada would indeed have a perfectly sensible solution to the question, but it's not necessarily the one Fortran should use. It turns out that if Ada does indeed require that the uses of the shortcut operators (Ada calls them control forms) must parenthesize for clarity, which is indeed a sensible solution. It's also one of the possibilities that occurred independently. Maybe Fortran should follow suit, but at present the proposed feature solves the problem differently. At any rate, the answer hardly needs to concern the Ada newsgroup anymore, so I for one will not cross-post any of the further discussion.

Moving from Ada 83 to Ada 95

From: Jeffrey Carter <jrcarter@acm.org>
Date: Thu, 25 Nov 2004 01:30:00
Subject: Re: Moving from Ada 83 to Ada 95
Newsgroups: comp.lang.ada

vrenna wrote:

> Has someone done such thing? I need to move a huge application made in Ada 83 and installed on Solaris, to Ada 95 (probably on Linux)... looks very tough but maybe there're automatic tools and good advices out there. What do you say?

I've moved large quantities simply by recompiling. Actual incompatibilities in the languages are quite rare. If the code is designed for portability, then I'd be surprised if you have a problem.

[Reports on industrial experiences of language migration are posted under: http://www.cs.kuleuven.ac.be/~dirk/papers -- su]

From: Christoph Karl Walter Grein <AdaMagica@web.de>
Date: Thu, 25 Nov 2004 08:44:50 +0100
Subject: Re: Moving from Ada 83 to Ada 95
Newsgroups: comp.lang.ada

I've moved a 800kLOC program from Ada83 running on Motorola 68040 to Ada95 running on IRIX, different compiler vendors, in a few weeks. Mostly an easy job - getting it to compile and run was a few days. What took the rest of the time was to find causes of crashes and non-portable code.

Problem areas:

Original code used overlays (for A use at X) for low level hardware access. If you do such erroneous (in Ada83) things, take care that alignments match or else you will get segmentation error crashes.

Unchecked_Conversion often had different sizes on Source and Target. This leads to non-portability and funny results.

Additionally you have to be careful with alignment issues (RM 13.9(7)). GNAT however does the correct thing, even if alignments do not match.

Software Engineering Ethics

From: Nick Roberts <nick.roberts@acm.org>
Date: Sat, 8 Jan 2005 04:08:55
Subject: Re: Software Engineering Ethics
Newsgroups: comp.lang.ada

Jeffrey Carter wrote:

> I've seen some ads for a position converting an Ada application to C++. They say they'll train you in C++, as if that were some sort of attraction. Principle 3 of the ACM/IEEE-CS Software Engineering Code of Ethics says, "Software engineers shall ensure that their products and related modifications meet the highest professional standards possible." Principle 6 says, "Software engineers shall advance the integrity and

> reputation of the profession."
> Isn't anyone who takes this job automatically in breach of these principles?

Not automatically, I think. They probably are, in actuality, but the problem is proving it.

I'm pretty sure the most effective way to evangelise the world about the advantages of Ada (and about the problems with C++ and its siblings) is to demonstrate to software businesses -- and their customers -- that using Ada is cheaper (in the long term) and helps make the product better.

From: Alexander E. Kopilovich <aek@VB1162.spb.edu>
Date: Sat, 8 Jan 2005 21:14:43 +0300 (MSK)
Subject: Re: Software Engineering Ethics
Newsgroups: comp.lang.ada

There is a faith-based ethics and a caste-based ethics.

Principle 3, which you mentioned, reflects faith-based approach to ethics, while Principle 6 reflects caste-based approach to ethics.

For an individual, participation in porting from Ada to C++ may (or may not) constitute some breach of Principle 3, but at the same time it may count as his/her increased support for Principle 6. Therefore, even for informed (and not too hungry) individual this is a matter of rather subtle choice.

And after all there are programmers/software engineers who are able to do porting from Ada to C++ without a breach of any reasonable ethics... just because they understand the matters deeply enough. (We have here in c.l.a. at least one obvious example of such a person.)

From: Cesar Rabak <crabak@acm.org>
Date: Fri, 07 Jan 2005 23:36:00 -0200
Subject: Re: Software Engineering Ethics
Newsgroups: comp.lang.ada

If porting code from one language to another would break these principles, we would be in mess ;-)

OTOH, if this post is trying to make a judgement of the relative merits of the two technologies/languages, you start to skate in thin ice.

IMNSHO best use of this energy could be try to contact the ads' responsible and understand why this movement is being made.

Ada Advocacy - Team-Ada

From: David Botton <david@botton.com>
Date: Wed, 24 Nov 2004 19:56:59 -0500
Subject: Ada Advocacy - Team-Ada
Newsgroups: comp.lang.ada

For those of you that are not familiar there is an e-mail list called Team-Ada (see the Ada FAQ - http://www.adapower.com/faq - but to make things short you join by sending the word subscribe to team-ada@acm.org) where Ada advocacy is discussed.

I've started this evening on the list the first of what I hope will be many of my suggestions to Juice Up Ada (There was certainly tons of Juice flowing when the Juice was on trial, perhaps we can get some going with Ada a bit on trial).

I've posted the following (Please join and respond on Team-Ada@acm.org for this one):

Welcome to the Team-Ada group therapy session #1

Suggestion #1:

A good article written not for academics or professionals, but to newbies that are just getting in to or interested in programing in general as to why Ada should be the language they embrace and learn first (We all know not to learn Basic first ;-).

- Think, if you were a student, would you want to learn anything but Java or C++ given the current marketplace.

- Most would rather learn almost any language in the world rather than Ada, COBOL or Fortran....

Personally, I started by doing stuff like looking up on a chart of 8080 instructions and poking the values in memory using trs-80 basic after converting from hex to decimal, but I am a certified freak who had nothing better to do when all his friends had Apple and Commodore, and was nine years old and not thinking about a job, the future, etc... Others are another matter:

So, the AdaPower challenge for the night - Why should any one learn Ada as their first language?

Let's go, whose in for a post :-)

Delphi and GNAVI

From: Brian May <bam@snoopy.apana.org.au>
Date: Sun, 09 Jan 2005 15:08:19 +1100
Subject: Re: Software Engineering Ethics
Newsgroups: comp.lang.ada

A job agency that contacted me recently saw on my resume that I had been involved in a Ada project. The representative said "That must be really old...". I said "No, it's relatively recent". No response. Either he realized he made a mistake or he considered me a total idiot for writing new software with an "obsolete" language. Not sure which one, although I suspect the later.

I think many people treat Ada as an obsolete language, much like Fortran (for example), without considering if this really is the case. Old doesn't always mean obsolete, not even with the rapid progress made in IT.

The ironic thing is (to the best of my knowledge), Borland still support Delphi, which is based on Pascal, and Ada is based on Pascal... I haven't heard anyone complain about Pascal being obsolete.

From: Martin Krischik <martin@krischik.com>
Date: Sun, 09 Jan 2005 11:24:01 +0100
Subject: Re: Software Engineering Ethics
Newsgroups: comp.lang.ada

But they renamed it to "Delphi" to get rid of "old memory" and bundled it with a rappid application development suite.

From: Jeffrey Carter <jrcarter@acm.org>
Date: Sun, 09 Jan 2005 19:27:04 GMT
Subject: Re: Software Engineering Ethics
Newsgroups: comp.lang.ada

I've heard Pascal called obsolete. For many users, the Pascal in Delphi is hidden, and even if they look at the code, it's called Delphi, not Pascal.

From: David Botton <david@botton.com>
Date: Sun, 9 Jan 2005 16:55:03 -0500
Subject: Re: Software Engineering Ethics
Newsgroups: comp.lang.ada

That is one of the goals of the GNAVI project (http://www.gnavi.org) you program it in GNAVI ;-)

David Botton

http://www.gnavi.org - Open Source Visual RAD

From: David Botton <david@botton.com>
Subject: Re: Software Engineering Ethics
Date: Thu, 13 Jan 2005 15:47:19 -0500
Newsgroups: comp.lang.ada

>> OK, but it needs a catchier (and pronounceable) name. :-)

> Starts out like gnave and rhymes with Navy. :-)

Actually I prefer to pronounce it: (silent G) N ah V ee - or sometimes pronouncing the g but still the same N ah V ee

Navi in Hebrew is prophet. Sort of a play on Delphi :-)

There is a functional (but certainly needing more work) IDE now and GUI builder and of course GWindows has been solid for years on Win32. Once the IDE hits 1.0 in a few months or so I'll start work on a marketing package, etc.

From: Nick Roberts <nick.roberts@acm.org>
Date: Thu, 13 Jan 2005 18:06:07
Subject: Re: Software Engineering Ethics
Newsgroups: comp.lang.ada

> OK, but it needs a catchier (and pronounceable) name. :-)

I'm not sure that it would be worthwhile changing the name now, but the project seems to lack a really bold, catchy logo. Ideas, anyone?

[See also "GNAVI Progress" and "GUI Programming for Beginners" in this issue -- su]

Ariane 5 FAQ

From: Alexander E. Kopilovich <aek@VB1162.spb.edu>
Subject: Ariane5 FAQ, Observer's version, 8th draft
Date: 2004-11-27 14:17:28 PST
Newsgroup: comp.lang.ada

Q. Was the Ada language somehow related to Ariane 5 crash in 1996?

A. Yes, at least some components of the Ariane 5 software was written in Ada language.

Q. Did that software cause the crash?

A. Yes and No. They simply put the software written for previous model -- Ariane 4 (where it worked well) -- to new Ariane 5, and did not bother themselves with testing it on the new rocket before the launch. So, when the Ariane 4 software appeared (in the flight) incompatible with new Ariane 5 they became very surprised -- and blamed the software.

Q. But media told us that there was an error in the software that caused that crash. Is it right?

A. No, it is wrong. There was no such an error in the software. The software worked perfectly for the purpose, for which it was created, that is, for Ariane 4. The software was not created for Ariane 5, and there were no reasons to expect that it should work for this new rocket. So, the error, which caused the crash was blinded use of a software created for another job. And this error was severely aggravated by subsequent error -- skipping mandatory test procedure before the first flight.

Q. But why on earth they expected that it should work if they have no reasons for it? Are you implying that they were idiots? (No conspiracy theories please.)

A. No. There was an unfortunate collision of popular expectations about modern high-tech devices with real difficult issues of international collaboration in sensitive technologies.

Ariane 5 was an international project (within ESA = European Space Agency), and at the same time it naturally belonged to an area of high secrecy (which is, as you probably know, traditionally maintained within strictly national frame). This created a difficult issue and caused uncommonly massive involvement of persons with political, diplomatic, economical etc. rather than technical background and/or experience into the high management of the project.

Those persons naturally have mostly consumer-like expectations about modern high-tech devices. This means that while they may be generally smart and able to occupy some position within large technical project, nevertheless they have different (from an engineer) default assumptions for many technical issues.

So they dealt with one critical part of the equipment as if it was some regular consumer market product from a reliable vendor: they assumed that they may use the device in all circumstances that aren't explicitly and clearly prohibited in its documentation. Because of their insufficient engineering background and/or experience they weren't aware of the difference in this respect between a complete product and its component part -- they did not know well enough that for the latter the defaults are opposite, that is, you should not use the component device in any circumstances that aren't explicitly and clearly allowed.

Q. But certainly there were engineers also, who can see possible consequences of that approach. So why they weren't alarmed enough?

A. This is difficult question indeed. An explanation exists, which tells that the informational paths within the project were interspersed with those managers of non-engineering kind, and because of that no one of the engineers can obtain enough information for recognition of the danger.

A contributing factor was the specifics of communications and crossings of responsibilities, which often manifests itself within international projects. Here is an insider's view on that specifics:

"As with many international projects, some of the information is eyes only. This is sometimes a burden for engineers that write the software, since they have to rely on good will and reliable deliveries of sub-components. As you can imagine, Ariane is a fairly complex system which relies on many "sub-systems"; now imagine that all those subsystems come from a different supplier. The integration of all of them is a very large and complex project on is own."

Q. Still don't understand how they managed to avoid testing?

A. They did not entirely avoid testing. Actually they tested most of the rocket equipment, except of the Inertial Reference System (which then caused the crash). This device was excluded from the test procedure and replaced by its simulator (for financial and perhaps schedule reasons). The simulator was written within Ariane 5 project. The crucial thing was that the developers were not given the documentation for the software, but source code only. By that administrative restriction some limitations of the software (which were clearly stated in the documentation) were obscured from the developers of the simulator. As a result, the simulator worked differently from the real device. (It helped to test other equipment, but no more -- the real device remained untested for the new rocket.) Subsequently, after the crash, the original programmers of the Ariane 4 device were blamed for not stating the limitations by comments within the source code (additionally to the documentation).

Q. So, if the limitations were clearly reflected in comments within the source code then most probably they would be seen by the simulator's developers and the disaster would be averted?

A. Probably No. Because simulation of the alignment function of real device was excluded from the contract for the simulator development. Consequently, the simulator's developers have no stimulus (without the documentation, which was not given to them) to look into the part of the source code where the limitations were violated in the flight. (They might have looked there out of curiosity, but time pressure and general stress surrounding the project left too little room for curiosity.)

The reason for omission of the alignment function from the simulator was that for Ariane 5 that function is not needed after takeoff, and that before takeoff that function was really identical for the Ariane 4 and Ariane 5. What was overlooked is that for the Ariane 4 that function WAS executed after takeoff (about 40 seconds), so the unchanged real device would execute that function for the Ariane 5 despite the absence of any need for it there.

Q. Can you explain in several words what was the actual cause of the launch failure, technically?

A. There are several points which are different for Ariane 5 vs. Ariane 4, one of which was instrumental to the events: Ariane 4 is a vertical launch vehicle where as Ariane 5 is slightly tilted. Ariane 4 software was developed to tolerate certain amount of inclination but not as much as required by Ariane 5. The chain of events was as follows:

- The on-board software detects that one of the accelerometers is out of range (actually, there was FPU exception generated when float-to-integer conversion exceeded the capacity of the integer), this was interpreted as hardware error and caused the backup processor to take over;

- The backup processor also detects that one of the accelerometers is out of range (the same way), which caused the system to advice an auto destruction.

Q. Where can I find official report for the investigation of the Ariane 5 crash?

A. At the moment of writing this FAQ this report was, for example, at: http://www.dcs.ed.ac.uk/home/pxs/Book/ariane5rep.html

But read it to the end, because your overall impression will probably be different (and wrong) if you stop in the

middle of it, deciding that you got it all clear enough.

Q. Where this topic was discussed in depth?

A. For example, in comp.lang.ada newsgroup (several times). Search that newsgroup for "Ariane 5", and you'll find several threads discussing this topic (most recent at the moment of starting this FAQ was quite long thread with subject line "Boeing and Dreamliner"; during the development of this FAQ another long thread with the subject line "Ariane5 FAQ" was running).

From: Mike Silva <snarflemike@yahoo.com>
Date: 25 Nov 2004 10:28:24 -0800
Subject: Re: Would You Fly an Airplane with a Linux-Based Control System?
Newsgroups: comp.lang.ada

David Botton <david@botton.com> wrote in message news:<20041122182920168 07%david@bottoncom>...

> For a real understanding of the Ariane 5 event, see the Ada FAQ: http://www.adapower.com/index.php?Command=Class&ClassID=FAQ&CID=328

A small but, I think, important correction. The hardware at the center of the failure was apparently built around the Motorola 68020/68881 chips, not the MIL-STD-1750. The "Operand Error" that triggered the failure is a hardware exception generated by the FPU when, among other conditions, a float-to-integer conversion exceeds the capacity of the integer, exactly as occurred. The reason this is important is because it shows that the exception was not generated by the Ada compiler code but by the hardware, and would therefore have occurred regardless of the programming language used. If that's the case then the "it wouldn't have exploded if it were written in C" argument evaporates, unless they want to argue that the exception handler behavior would have been specified differently if the implementation language was C -- not likely!

From: Alex R. Mosteo <devnull@mailinator.com>
Date: Fri, 26 Nov 2004 11:11:26 +0100
Subject: Re: Would You Fly an Airplane with a Linux-Based Control System?
Newsgroups: comp.lang.ada

> (...) If that's the case then (…) -- not likely!

Anyway, isn't the C argument ridiculous? I mean, is preferable to have a huge explosive thing flying on corrupted data? So it can by chance go where it should... or not?

I know[*] in this particular case these component had no further purpose in the flight so it would have get away safely... but that's not relevant IMO.

[*] (IIRC it was only used during some limited time after lift-off).

From: Adrien Plisson <aplisson-news@stochastique.net>
Date: Fri, 26 Nov 2004 14:40:29 +0100
Subject: Re: Would You Fly an Airplane with a Linux-Based Control System?
Newsgroups: comp.lang.ada

> Anyway, isn't the C argument ridiculous?

Well, it depends on the interpretation you make of it.

It may be an argument against Ada and for C: 'look, the bad Ada language made the whole thing crash, whereas the good C language would have made it fly' (where? the argument does not tell, but surely not on the original path); but it can also be interpreted as an argument against C and for Ada: 'look, the cool Ada language prevented the whole thing to get out of control, whereas the bad C language would have continue to fly it without notice'

So, the ridiculousness of the argument depends on the interpretation, and I really think you are misinterpreting there.

From: Enrique Laso Leon <enrique.laso-leon@tele2.fr>
Date: Sat, 4 Dec 2004 19:58:07 +0100
Subject: Re: Would You Fly an Airplane with a Linux-Based Control System?
Newsgroups: comp.lang.ada

I am wondering why people try to make this accident an issue with the programming language and not what it was : a total failure in a software project management.

The problem here was that the people who designed the IRS for Ariane 4 used an assumption on its trajectory in order to avoid a check that would have made the software tolerant to Ariane 5 trajectory (but why ?). This is at best ignoring a basic rule of engineering: expect your design to be used in a way you did not think about, because this is just what is going to happen. It applies to machinery as it applies to software. How many of us use "bugs" or "safety flaws" in our favorite applications in order to get things done?

The other problem was with the baffling lack of testing. Once more it comes from a management belief that experimentation is the root of all evil, for it takes time thus money. Engineers there have much responsibility for the existence of this belief. We tend to sell as a strong point that our design and analysis methods are so perfect that we can produce zero fault out of the box. This simply forgets: that engineers, even when supported by the most efficient methods and computing tools, are human beings; that systems are getting more complex than anything a human organisation can cope with; and that error is not only probable, it is frequent; etc.

Tackle those two issues and you avoid blowing up a brand new rocket and 4 satellites. Regardless of the programming language.

From: Rod Haper <rhaper@houston.rr.com>
Date: Sat, 27 Nov 2004 00:55:34
Subject: Re: Would You Fly an Airplane with a Linux-Based Control System?
Newsgroups: comp.lang.ada

Marius Amado Alves wrote:

> Mike Silva wrote:

>> What was the bug? Since there wasn't one, your answer should prove interesting!

> Did they fix the hardware or the software? The inevitable conclusion from your answer should prove interesting!

Butting into this eternal argument:

The "bug" that got "fixed" was the specification. That in turn necessitated a change to the software to comply with the updated specification. The "error" was in the old Ariane IV's specification's lack of applicability to the new Ariane V's requirements. The "failure" was one of design, not software implementation, and was independent of what language was or might have been used for the implementation.

What is your point vis-à-vis hardware or software? The "conclusion" I draw is that you seem to be hung up on some agenda which ignores the simple facts of the case.

[See also "DDC-I - Riding High with Cassini-Huygens" and "Cassini-Huygens Reachs Titan" in this issue -- su]

GUI Programming for Beginners

From: munnoch <munnoch@btinternet.com>
Date: Tue, 16 Nov 2004 19:25:04
Subject: GUI programming --a hopeful newbie =)
Newsgroups: comp.lang.ada

I was wondering if any of you guys could point me to sites/books/etc. with information about making a GUI (for Windows) for an Ada program?

PS, as stated in the Subject, I'm a newbie at this, so I'll welcome comments regarding my need for more practise with Ada before trying a GUI if that would be beneficial =)

From: David Botton <david@botton.com>
Date: Thu, 18 Nov 2004 08:55:39 -0500
Subject: Re: GUI programming --a hopeful newbie =)
Newsgroups: comp.lang.ada

As GWindows is going toward multi platform now (already started on Mac OS

X), if you are looking for Windows now and Linux and Mac later, GWindows is a superior solution as it offers a true native binding to the host OS's toolkits.

It also happens to be extremely easy to use even for a beginner, the GNAVI ICG already allows complex GUIs to be created and worked on easily with XML specs (the GNAVI IDE / GUI Builder is being built with it).

From: David Botton <david@botton.com>
Date: Thu, 18 Nov 2004 09:19:24 -0500
Subject: Re: GUI programming --a hopeful newbie =)
Newsgroups: comp.lang.ada

Bernd Specht wrote:

> in this case I would suggest, that you first start with console programming until you get some experience with programming in general and the Ada language in special.

I disagree. There is no reason that a first class in Ada should not be:

```
with GWindows.Message_Boxes;
use  GWindows.Message_Boxes;
procedure Hello_World is
begin
   Message_Box ("My_App",
"Hello World!");
end Hello_World;
```

and a Second:

I am planning a series of on-line videos to teach GNAVI / Ada programming and intend on doing so.

Unless you have a captured audience being forced to learn Ada, its time to start thinking out of the box about how to influence people to want to learn Ada!

> I think it would be too much at a time if you want to learn all at once: programming, Ada, *and* a GUI system. Maybe you will become frustrated soon.

Certainly not if you are using higher level frameworks like GWindows, CLAW or JEWL (and to some degree GtkAda).

I find trying to work with Ada.Text_IO more complex to use or teach than GUI programming with any of the above ;-)

> Switch to GUI development if you've got some experience.

Start with it. No reason to live in a DOS / UNIX prompt world in 2004.

> If you want concentrate on Windows *and* Ada then look at GWindows.

Look at it any ways :-)
http://www.gnavi.org/gwindows

> If you think about writing (Windows-) GUI not only with Ada but maybe with Pascal or C, too, then look at the pure Win32-API which is used very similar with these other languages.

Look at learning GWindows and understanding the source. If you are going to program in C, you will want to use the techniques there. If you are going to use Pascal, than forget Delphi and get with GNAVI.

> If you think of (maybe one day in the far future) writing for Windows *and* Linux, then have a look at GTK which allows you to write portable GUIs.

GWindows will be on Linux in time, but GtkAda is a good solution if your needs are now.

From: David Botton <david@botton.com>
Subject: Re: GUI programming --a hopeful newbie =)
Date: Thu, 18 Nov 2004 08:50:35 -0500
Newsgroups: comp.lang.ada

You should take a look at GNAVI - http://www.gnavi.org

GNAVI is the Open Source answer to Visual Basic and Delphi

The full IDE is being created now and a working prototype will be available in the next few weeks.

In the interim it includes:

GWindows - An easy to use for beginners and highly extendable for the advanced, full featured binding to Win32 that is being ported to Mac OS X and Gtk

GNATCOM - For Active X control creation and use

GNAVI ICG - A tool to generate or modify a GWindows based application based on an XML spec

The Mac OS X port is in progress, the GTK port will likely start in a few months.

[See also "GNAVI Progress" and "Delphi and GNAVI" in this issue -- su]

Conference Calendar

This is a list of European and large, worldwide events that may be of interest to the Ada community. Further information on items marked ♦ is available in the Forthcoming Events section of the Journal. Items in larger font denote events with specific Ada focus. Items marked with ☺ denote events with close relation to Ada.
The information in this section is extracted from the on-line *Conference announcements for the international Ada community* at: http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programs, URLs, etc. and are updated regularly.

2005

April 02-08 **Conference on Design, Analysis, and Simulation of Distributed Systems** (DASD'2005), San Diego, California, USA. Theme: "Making simulation and analysis successful through novel distributed system design, methodologies, and management". Topics include: Distributed Systems, Design and implementation approaches for complex systems using Petri nets, Distributed real-time systems, Formal concepts and methods for validation and testing, High level architecture in distributed systems, etc.

April 02-10 **European Joint Conferences on Theory and Practice of Software** (ETAPS'2005), Edinburgh, Scotland, United Kingdom. Event includes: conferences from 4-8 April, 2005, satellite events on 2-3 and 9-10 April, 2005

- April 02-03 3rd **Workshop on Quantitative Aspects of Programming Languages** (QAPL'2005). Topics include: the design of probabilistic and real-time languages; of semantical models for such languages; the discussion of methodologies for the analysis of probabilistic and timing properties (e.g. security, safety, schedulability); applications to safety-critical systems; etc.
- April 02-10 8th **International Conference on Fundamental Approaches to Software Engineering** (FASE'2005). Topics include: Systematic approaches towards evolution management in large scale systems, continuous software engineering, and improvement and adaptation of legacy systems to altered requirements; Rigorous approaches to the design, testing, and maintenance of reactive, mobile, and distributed software systems; Integration of formal concepts and current best practices in industrial software development; Experience reports on best practices with development tools, software development kits, ...; etc.
- April 03 4th **International Workshop on Compiler Optimization Meets Compiler Verification** (COCV'2005). Topics include: optimizing and verifying compilation, and related fields such as translation validation, certifying and credible compilation, but also programming language design and programming language semantics.
- April 03 5th **Workshop on Language Descriptions, Tools and Applications** (LDTA'2005). Topics include: Program analysis, transformation, and generation; Formal analysis of language properties; Automatic generation of language processing tools.
- April 04-08 14th **International Conference on Compiler Construction** (CC'2005). CC is undergoing an expansion. Traditionally, CC has focused on compiler construction; CC now seeks to become a conference for research on a broader spectrum of programming tools, from refactoring editors to checkers to compilers to virtual machines to debuggers. Topics include: compilation techniques, incl. program representation and analysis, code generation and code optimization; run-time techniques, incl. memory management; compilation techniques for embedded code; compilers for parallel and distributed computing; compilation techniques for security and safety; design of novel language constructs and their implementation; software tools, incl. debuggers, profilers, code verifiers; etc.
- April 07-09 **International Symposium on Trustworthy Global Computing** (TGC'2005). Topics include: language-based security, reliability and business integrity, language concepts

and abstraction mechanisms, type checkers, software principles to support debugging and verification, etc.

☺ April 03-08 **International Parallel and Distributed Processing Symposium** (IPDPS'2005), Denver, Colorado, USA. Topics include: Applications of parallel and distributed computing; Parallel and distributed software, including parallel programming languages and compilers, operating systems, middleware, libraries, programming environments and tools; etc.

☺ April 04 10th **International Workshop on High-Level Parallel Programming Models and Supportive Environments** (HIPS'2005). Topics include: Concepts and languages for parallel and Grid programming (Component programming models, Refactoring of existing applications into components, Language and platform interoperability, Concurrent object-oriented programming, Extensions to traditional programming models, ...); Supportive techniques and runtime environments (Compiler techniques, ...); Tools for high-level parallel programming; etc.

April 04 3rd **International Workshop on Parallel and Distributed Systems: Testing and Debugging** (PADTAD'2005). Topics include: optimizing and verifying compilation, and related fields such as translation validation, certifying and credible compilation, but also programming language design and programming language semantics.

☺ April 05-08 2nd **Jahrestagung Fachbereich "Sicherheit - Schutz und Zuverlässigkeit"** (*2nd Annual Conference on Safety and Security*), Regensburg, Germany. Event includes: special session "Informationssicherheit im Automobil", workshop "Privacy Respecting Incident Management", etc. Contributions welcome in English or in German. Topics include (in German): Vertrauenswürdige Softwarekomponenten; Zuverlässigkeit und Fehlertoleranz in Hardware- und Softwaresystemen; Formale Techniken, Modellierung, Spezifikation und Verifikation; Betriebssicherheit unter extremen Bedingungen; Standards und Normung; Sicherheitsevaluation und -zertifizierung; Kosten von Sicherheit; etc.

April 06-08 **Software & Systems Quality Conferences** (SQC'2005), Düsseldorf, Germany. Event includes: ICSTEST International Conference on Software Testing, SQM, a congress focussing on Software Quality Management, and CSVHC Conference on Software Validation for Health Care.

April 10-13 **The Conference for Software Practice Advancement** (SPA'2005), Wyboston, Bedfordshire, England. Topics include: Languages; Distributed, component-based development; Pervasive or embedded systems; Patterns and pattern languages; Comparative experience (what we have learned or can learn from other disciplines); Lessons learned/experience reports; etc.

April 11 **2005 Ada-Belgium General Assembly**, ULB, Brussels, Belgium. Program includes: short product announcement by Patricia Langle (Aonix France) of an Eclipse plug-in for Ada (ObjectAda or GNAT); technical presentation by Jean-Pierre Rosen (Adalog, France) on “Web-enabling Ada Applications with AWS”.

April 11-13 12th **Annual European Concurrent Engineering Conference** (ECEC'2005), Toulouse, France. Topics include: engineering of embedded systems, specification languages, distributed computing environments, practical solutions, pitfalls and success stories, case studies, pilot projects and experiments, etc.

April 11-13 9th **International Conference on Empirical Assessment in Software Engineering** (EASE'2005), Keele University, UK. Topics include: Evaluation of products, components and services; Process and tool evaluation; Quality assessment; Software experiments, case studies and observational studies; etc.

☺ April 18 ICRA2005 - **Workshop on Principles and Practice of Software Development in Robotics**, Barcelona, Spain. Topics include: advanced software development techniques for building robotic systems.

April 18-20 17th Conference on Software Engineering Education and Training (CSEET'2005), Ottawa, Canada

April 18-21 Annual **Systems and Software Technology Conference** (SSTC'2005), Salt Lake City, Utah, USA.

☺ April 22 **Computer Languages for Scientific Computing**, Birmingham, UK. Topics include: specific features in languages that support scientific programming, etc.

♦ April 25 **2005 Ada-Spain Technical Meeting**, Madrid, Spain. Topics include (in Spanish): Next revision of the Ada language, Embedded systems, Distributed systems, Industrial experiences. After the meeting, Ada-Spain will hold its 17th General Assembly and the end of which the winners of the "Ada-Spain's 13th Contest for the best academic work in Ada" will be announced.

April 27-29 5th International SPICE Conference on Software Process Improvement and Capability dEtermination (SPICE'2005), Klagenfurt, Austria.

☺ April 30 **Mid-Atlantic Student Workshop on Programming Languages and Systems** (MASPLAS'2005), Newark, Delaware, USA. Topics include: Implementation of Language Features, Compiler Construction and Optimization Techniques, Languages and Compilers for Parallel Programming, Design of Programming Languages, Type Checking, Teaching Programming Techniques, etc.

May 02-06 **International Conference on Practical Software Quality and Testing** (PSQT'2005 West), Las Vegas, Nevada, USA. Theme: "Meeting the Software Security Challenge through Quality and Testing".

☺ May 15-21 27th **International Conference on Software Engineering** (ICSE'2005), St Louis, Missouri, USA. Topics include: Software architectures and design; Software components and reuse; Software security; Software safety and reliability; Reverse engineering and software maintenance; Software economics; Empirical software engineering and metrics; Distribution and parallelism; Software tools and development environments; Programming languages; Object-oriented techniques; Embedded and real-time software; etc.

- May 14-15 8th **International SIGSOFT Symposium on Component-Based Software Engineering** (CBSE'2005). Theme: "Software Components at Work". Topics include: Static analysis and execution monitoring of system properties; Components for real-time, secure, safety critical and/or embedded systems; etc.
- May 15-16 13th **IEEE International Workshop on Program Comprehension** (IWPC'2005). Topics include: Comprehension during large scale maintenance, reengineering, and evolution of existing systems; Reverse engineering for the purpose of program comprehension; etc.
- ☺ May 17 **Workshop on Architecting Dependable Systems** (WADS'2005). Topics include: all topics related to software architectures for dependable systems.
- May 17 3rd **Workshop on Software Quality** (WoSQ'2005). Topics include: Software Product Evaluation and Certification; Tradeoffs in Quality during software development; Software Quality Education; Methods and Tools for Quality Assurance; Software Quality at different stages of the development lifecycle; Building quality into software products; Testing, Inspections, Walkthroughs and Reviews; etc.

May 22-25 5th **International Conference on Computational Science** (ICCS'2005), Atlanta, USA. Theme: "Advancing Science through Computation". Topics include: Parallel and Distributed Computing, etc.

May 23-25 9th **Brazilian Symposium on Programming Languages** (SBLP'2005), Recife, PE, Brazil. Topics include: programming language design and implementation, formal semantics of programming languages, theoretical foundations of programming languages and teaching programming languages, etc.

May 27 – June 01 3rd **International Software Development Conference** (SWDC'2005), Reykjavik, Iceland. Topics include: Project success and failure analysis; Software project risk management; Software process improvement; etc.

☺ May 30 – June 02 **DAta Systems In Aerospace** (DASIA'2005), Edinburgh, Scotland, UK.

June 06-09 5th **International Conference on Application of Concurrency to System Design** (ACSD'2005), St Malo, France. Topics include: Correct-by-construction design methods and integration of verification techniques with the design process; etc.

June 06-10 25th **International Conference on Distributed Computing Systems** (ICDCS'2005), Columbus, Ohio, USA. Sponsored by The IEEE Computer Society Technical Committee on Distributed

Processing. Topics include: Fault Tolerance & Dependability, Middleware, Real-time & Embedded Systems, Security, Formal Verification, etc.

June 13-17 — 17th **Conference on Advanced Information Systems Engineering** (CAiSE'2005), Porto, Portugal. Topics include: Model and Software Reusability, Distributed and Open Architectures, Languages for IS, etc.

☺ June 14-17 — 34th **International Conference on Parallel Processing** (ICPP'2005), Oslo, Norway. Topics include: Compilers and Languages, Programming Methodologies, Tools, Parallel Embedded Systems, etc.

June 15-17 — 7th **IFIP International Conference on Formal Methods for Open Object-based Distributed Systems** (FMOODS'2005), Athens, Greece. In conjunction with DAIS'2005 (Distributed Applications and Interoperable Systems). Topics include: Formal techniques for specification, design or analysis; Verification, testing and validation; Component-based design; Type systems for programming languages; Formal models for security; Applications and experience, carefully described; etc.

June 15-17 — 5th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS'2005), Athens, Greece.

June 16-20 — 10th **IEEE International Conference on the Engineering of Complex Computer Systems** (ICECCS'2005), Shanghai, China. Topics include: Tools, environments, and languages for complex systems; Formal methods for developing complex systems; Software and system development processes for complex systems; Software review, inspection, and testing; Formal proof and model checking; Human factors and collaborative aspects; Interoperability and standardization; Systems and software safety and security; Industrial automation, embedded and/or real time systems; etc.

June 18-23 — 6th **International Conference on eXtreme Programming and Agile Processes in Software Engineering** (XP'2005), Sheffield, UK. Topics include: Case studies, experiments and practioner's reports; Scalability issues; Refactoring and continuous integration; Use of SW development tools and environments; etc. Deadline for early registration: May 7, 2005.

♦ June 20-24 — 10th **International Conference on Reliable Software Technologies - Ada-Europe'2005**, York, UK. Sponsored by Ada-Europe, in cooperation with ACM SIGAda.

June 27-29 — 10th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2005), Monte de Caparica, Portugal. Deadline for submissions: April 18, 2005 (working group membership application).

June 27-30 — 2005 International Multiconference in Computer Science and Computer Engineering (IMCSE'2005), Las Vegas, Nevada, USA.

- ☺ June 27-30 **International Conference on Programming Languages and Compilers** (PLC'2005).
- June 27-30 **International Conference on Software Engineering Research and Practice** (SERP'2005). Topics include: Formal methods in software engineering, Software engineering and high assurance systems, Software maintenance, Component-based software engineering, Quality-based software engineering, Real-time software engineering, Critical systems, Verification and validation, Software testing, Quality management, Object-oriented software engineering, Case studies, etc.

July 06-10 — 17th **International Conference on Computer-Aided Verification** (CAV'2005), Edinburgh, Scotland, UK. Topics include: Algorithms & tools for verifying models and implementations, Program analysis and software verification, Applications and case studies, Verification in industrial practice, etc.

☺ July 11-14 — **OMG Annual Workshop on Distributed Object Computing for Real-time and Embedded Systems**, Washington, DC, USA. Topics include: Real-time systems; Embedded systems; Fault-tolerant systems; High-availability systems; Safety-critical systems; Real-time middleware, including real-time CORBA; Modelling notations (including Unified Modelling Language, UML); High-level real-time programming models; etc.

July 11-15 — 32nd **International Colloquium on Automata, Languages and Programming** (ICALP'2005), Lisbon, Portugal. Topics include: Parallel and Distributed Computing; Principles of Programming

Languages; Formal Methods; Program Analysis and Transformation; Specifications, Verifications and Secure Programming; etc. Affiliated Workshops on July 9-10 and 16-17, 2005.

July 11-15 — 1st **International Conference on Open Source Systems** (OSS'2005), Genova, Italy. Topics include: Introduction of OSS in companies and Public Administrations, Empirical analysis of OSS, Case studies and experiments, etc.

July 17-20 — 24th **Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing** (PODC'2005), Las Vegas, Nevada, USA. Topics include: all areas of distributed systems; including: Distributed applications; Specification, semantics, and verification; Distributed middleware platforms; etc. Deadline for submissions: May 1, 2005 (nominations Edsger W. Dijkstra Prize in Distributed Computing).

July 18-22 — 13th **International Symposium of Formal Methods Europe** (FM'2005), Newcastle upon Tyne, UK. Topics include: introducing formal methods in industrial practice (technical, organizational, social, psychological aspects); reports on practical use and case studies (reporting positive or negative experiences); tool support and software engineering; environments for formal methods; etc. Deadline for submissions: May 09, 2005 (tools exhibition, demonstrations)

July 25-28 — 29th **Annual International Computer Software and Applications Conference** (COMPSAC'2005), Edinburgh, Scotland, UK. Theme: "High Assurance Software Systems". Topics include: Dependable service provision, Trustworthy software, Software safety, Software fault tolerance, High performance software, Component-based software, Design patterns, Software certification, Software standards, Software engineering education, Embedded systems, Middleware systems, Automotive telematics, etc.

☺ July 25-29 — 19th **European Conference on Object-Oriented Programming** (ECOOP'2005), Glasgow, Scotland, UK. Topics include: Concurrent, real-time and parallel systems; Design patterns; Distributed systems; Frameworks and software architectures; Language design and implementation; Programming environments; Adaptability; Formal methods; Software evolution; etc. Deadline for submissions: May 6, 2005 (demos, posters, exhibitions).

- ☺ July 26 — **Workshop on Practical Problems of Programming in the Large** (PPPL'2005). Topics include: The role of the software-architect in the phases requirements engineering, software design and development; Negative results: what went wrong although it should have worked according to software engineering folklore; Keeping systems with large amounts of classes / objects / modules / components organised; Refactoring, software evolution & migration; etc. Submission deadline: 6 May 2005.

August 29 – Sept. 02 — 13th **IEEE International Requirements Engineering Conference** (RE'2005), Paris, France. Deadline for submissions: April 28, 2005 (doctoral symposium, posters, research demonstrations)

☺ Aug. 30 – Sept. 02 — 11th **International Conference on Parallel and Distributed Computing** (Euro-Par'2005), Lisboa, Portugal. Topics include: Support Tools and Environments; Scheduling and Load Balancing; Compilers for High Performance; Distributed Systems and Algorithms; Parallel Programming: Models, Methods, and Languages; etc.

August 30 – Sept. 03 — 31st **EUROMICRO Conference on Software Engineering and Advanced Applications** (EUROMICRO'2005), Porto, Portugal. Topics include: Component-Based Software Engineering, Software Process and Product Improvement, Component Models for Dependable Systems, Value-based Software Engineering, etc.

☺ September 11-14 — **Workshop on Language-Based Parallel Programming Models** (WLPP'2005), Poznan, Poland. Topics include: Language and library implementations; Proposals for, and evaluation of, language extensions; Applications development experiences; Comparisons between programming models; Compiler Implementation and Optimization; etc. Deadline for submissions: April 31, 2005.

☺ September 12-14 — 18th **International Conference on Parallel and Distributed Computing Systems** (PDCS'2005). Las Vegas, Nevada, USA Topics include: Parallel and Distributed Systems Software; Languages, Compilers and Operating Systems; Libraries and Programming Environments; Message Passing and Distributed Shared Memory Paradigms; Software Development, Services, Support, and Tools; Middleware for Parallel and Distributed Computing; Embedded Systems; Parallel and Distributed Applications; etc. Deadline for paper submissions: April 8, 2005.

☺ September 13-16 **International Conference on Parallel Computing** 2005 (ParCo2005), Malaga, Spain. Topics include: applications; software engineering methodologies, methods and tools for developing and maintaining parallel software, incl. parallel programming models and paradigms, development environments, languages, compiling and run-time tools; etc. Deadline for submissions: July 31, 2005 (draft full papers).

September 19-22 11th **International Software Metrics Symposium** (Metrics'2005), Como, Italy. Topics include: Effort and cost estimation; Defect rate and reliability prediction; Quality Assurance; Empirical studies of global software development projects, open source software projects, agile development projects; etc. Deadline for submissions: April 1, 2005 (workshops, dissertation forum), May 30, 2005 (industry track).

September 19-23 9th Int'l IEEE Enterprise Distributed Object Computing Conference (EDOC'2005), Enschede (NL).

September 25-30 21st **IEEE International Conference on Software Maintenance** (ICSM'2005), Budapest, Hungary. Topics include: issues related to maintaining, modifying, enhancing, and testing operational systems, and designing, building, testing, and evolving maintainable systems. Deadline for submissions: April 30, 2005 (industrial applications, panels, tool demonstrations, dissertation forum, tutorials).

September 26-30 3rd **World Conference for Software Quality** (3WCSQ), Munich, Germany. Topics include: Software Construction, Integration and Testing, Verification and Validation, Risk Management and Problem resolution, Training and Education, Maintenance and Customer Support, Reliability Engineering, Embedded Systems, Medical Devices, Automotive and Automation, Avionics and Transportation Systems, etc.

October 02-07 8th **International Conference on Model Driven Engineering Languages and Systems** (MoDELS'2005), Montego Bay, Jamaica. Formerly the UML series of conferences. Topics include: Model-driven development methodologies, approaches, and languages; Empirical studies of modeling and model-driven development; Tool support for any aspect of model-driven development or model use; Semantics of modeling languages; etc. Deadline for submissions: April 4, 2005 (experience and scientific full papers), May 6, 2005 (workshops), May 20, 2005 (educators' symposium), June 6, 2005 (tutorials), TBA (doctoral symposium, tools and exhibits, posters and demos).

☺ October 16-20 20th **Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications** (OOPSLA'2005), San Diego, California, USA. Sponsored by ACM SIGPLAN in cooperation with SIGSOFT. Deadline for submissions: June 1, 2005 (Onward! Films, Onward! Presentations), July 1, 2005 (Demonstrations, Posters, Doctoral Symposium Submissions, Student Research Competitions papers).

October 25-26 **International Conference on Software Testing** (ICSTEST-E'2005), Bilbao, Spain. Topics include: Transportation and Safety-Critical Systems, Industry real experiences, Verification and Validation, Techniques for real time systems, Static and Dynamic analysis, Norms and standards, etc. Deadline for submissions: April 15, 2005

☺ Oct. 31 – Nov. 04 7th **International Symposium on Distributed Objects and Applications** (DOA'2005), Agia Napa, Cyprus. Topics include: Application case studies of distribution technologies; Design patterns for distributed systems; Distribution technologies for embedded systems; Interoperability between object systems and complementary technologies; Real-time solutions for distributed objects; Scalability for distributed objects and object middleware; Security for distributed object systems; Specification and enforcement of Quality of Service; Technologies for reliability and fault-tolerance; etc. Deadline for submissions: May 24, 2005 (abstracts), May 31, 2005 (papers).

November 01-04 7th **International Conference on Formal Engineering Methods** (ICFEM'2005), Manchester, UK. Topics include: all aspects of formal engineering methods, from theoretical work that promises various benefits, to application to real production systems. Deadline for submissions: May 20, 2005.

♦ November 13-17 2005 **ACM SIGAda Annual International Conference** (SIGAda'2005), Atlanta, Georgia, USA.

November 29 – Dec. 02 5th **International Conference on Integrated Formal Methods** (IFM'2005), Eindhoven, The Netherlands. Deadline for submissions: May 18, 2005

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

Rationale for Ada 2005: 1 Object oriented model

John Barnes

John Barnes Informatics, 11 Albert Road, Caversham, Reading RG4 7AN, UK; Tel: +44 118 947 4125; email: jgpb@jbinfo.demon.co.uk

Abstract

This paper describes various important improvements to the object oriented model for Ada 2005.

First an alternative more traditional prefixed notation for calling operations has been introduced. A major improvement is that Java-like interfaces are introduced thereby permitting simple multiple inheritance; null procedures have also been introduced as a category of operation. Greater general flexibility is provided by allowing type extension at a more nested level than that of the parent.

There are also explicit features for overcoming nasty bugs which arise from confusion between overloading and overriding.

Keywords: rationale, Ada 2005.

1 Overview of changes

The WG9 guidance document [1] identifies very large complex systems as a major application area for Ada. It says

"The main purpose of the Amendment is to address identified problems in Ada that are interfering with Ada's usage or adoption, especially in its major application areas (such as high-reliability, long-lived real-time and/or embedded applications and very large complex systems). The resulting changes may range from relatively minor, to more substantial."

Object oriented techniques are of course important in very large systems in providing flexibility and extensibility. The document later asks the ARG to pay particular attention to

B Improvements that will remedy shortcomings in Ada. It cites in particular improvements in OO features, including adding a Java-like interface feature and improved interfacing to other OO languages.

Ada 2005 does indeed make many improvements in the object oriented area. The following Ada Issues cover the relevant changes and are described in detail in this paper:

218 Accidental overloading when overriding

251 Abstract interfaces to provide multiple inheritance

252 Object.Operator notation

260 Abstract formal subprograms & dispatching constructors

284 New reserved words

310 Ignore abstract nondispatching ops during overloading

344 Allow nested type extensions

348 Null procedures

391 Functions with controlling results on null extension

396 The "no hidden interfaces" rule

400 Wide and wide-wide images

401 Terminology for interfaces

405 Progenitors and `Ada.Tags`

407 Terminology and semantics for prefix names

411 Equality for types derived from interfaces

These changes can be grouped as follows.

First we discuss the fact that Ada 2005 has three new reserved words, **interface**, **overriding**, and **synchronized**. It so happens that these are all used in different aspects of the OO model and so we discuss them in this paper (284).

Then there is the introduction of the `Obj.Op` notation used by many other languages (252, 407). This should make Ada easier to use, improve its image, and improve interfacing to other languages.

A huge improvement is the addition of Java-like interfaces which allow proper multiple inheritance (251, 396, 401, 411). A related change is the introduction of null procedures as a category of operation somewhat like abstract operations (348).

Type extension is now permitted at a more nested level than that of the parent type (344). An important consequence is that controlled types no longer need to be declared at library level.

An interesting development is the introduction of generic functions for the dynamic creation of objects of any type of a class (260, 400, 405). These are sometimes called object factory functions or just object factories.

Additional syntax permits the user to say whether an operation is expected to be overriding or not (218). This detects certain unfortunate errors during compilation which otherwise can be difficult to find at execution time. A small change to the overriding rules is that a function with a controlling result does not "go abstract" if an extension is in fact null (391). Finally, we discuss a minor but useful change to the overloading rules; in a sense this is not about OO at all since it concerns the rules for nondispatching operations but it is convenient to discuss it here (310).

There are in fact many other OO related improvements in Ada 2005 concerning matters such as access types, visibility, and generics. They will be described in later papers.

2 Reserved words

Ada 2005 has three further reserved words namely **interface**, **overriding**, and **synchronized**. Readers may recall that Ada 95 had six more reserved words than Ada 83 and the fact that this meant that some programs were incompatible and thus had to be rewritten loomed large in the minds of many commentators.

When new syntax for the introduction of interfaces was being discussed it was strongly felt that incompatibilities should be avoided and that any new syntax words should be unreserved. It was also noted that Interface was a popular identifier and that making it a reserved word would cause many programs to have to be rewritten.

However, it was soon realised that treating Interface as unreserved would have permitted sequences such as

```
type T is interface;
subtype Interface is T;
```

in which Interface is a subtype of the interface T. This would have been total madness. Some reviewers also had memories of PL/I in which words such as IF were not reserved so that one could write IF IF ... where the first IF is a syntax word and the second is a user identifier.

Accordingly it was decided that the new words would have to be reserved. No sensible alternative to **interface** could be thought of although it would be irritating for users who had packages called Interface -- actually a brief survey revealed that most such packages had longer names such as Radar_Interface so that the problem was more apparent than real. The other new reserved words **overriding** and **synchronized** clearly present less of a problem since they are less likely to have been used as identifiers.

3 The prefixed notation

As mentioned in the Introduction, the Ada 95 object oriented model has been criticized for not being really OO since the notation for applying a subprogram (method) to an object emphasizes the subprogram and not the object. Thus given

```
package P is
  type T is tagged ... ;

  procedure Op(X: T; ... );
  ...
end P;
```

then we usually have to write

```
P.Op(Y, ... );              -- subprogram first
```

in order to apply the operation to an object Y of type T whereas an OO person would expect to write

```
Y.Op( ... );                -- object first
```

Some hard line OO languages such as Smalltalk take the view that everything is an object and that all activities are operations upon some object. Thus adding 2 and 3 can be seen as sending a message to 2 instructing 3 to be added to it. This is clearly an extreme view.

Older languages take the view that subprograms are dominant and that they act upon parameters which might be raw numbers such as 2 or denote objects such as a circle. Ada 95 primarily takes this view which reflects its Pascal foundation over 20 years ago. Thus if Area is a function which returns the area of a circle then we write

```
A := Area(A_Circle);
```

However, when we come to tasks and protected objects Ada takes the OO view in which the identity of the object comes first. Thus given a task Actor with an entry Start we call the entry by writing

```
Actor.Start( ... );
```

So Ada 95 already uses the object notation although it only applies to concurrent objects such as tasks. Other objects and, in particular, objects of tagged types have to use the subprogram notation.

A major irritation of the subprogram notation is that it is usually necessary to name the package containing the declaration of the subprogram thus

```
P.Op(Y, ... );              -- package P mentioned
```

There are two situations when P need not be mentioned -- one is where the procedure call is actually inside the package P, the other is where we have a use clause for P (and even that sometimes does not give the required visibility). But these are special cases.

In Ada 2005 we can replace P.Op(Y, ...); by the so-called prefixed notation

```
Y.Op( ... );                -- package P never mentioned
```

provided that

- T is a tagged type,
- Op is a primitive (dispatching) or class wide operation of T,
- Y is the first parameter of Op.

The reason there is never any need to mention the package is that, by starting from the object, we can identify its type and thus the primitive operations of the type. Note that a class wide operation can be called in this way only if it is declared at the same place as the primitive operations of T (or one of its ancestors).

There are many advantages of the prefixed notation as we shall see but perhaps the most important is ease of maintenance from not having to mention the package containing the declaration of the operation. Having to name the package is often tricky because in complicated situations involving several levels of inheritance it may not be obvious where the operation is declared. This happens especially when operations are declared implicitly and

when class-wide operations are involved. Moreover if we change the structure for some reason then operations might move.

As a simple example consider a hierarchy of plane geometrical object types. All objects have a position given by the two coordinates *x* and *y* (this is the position of the centre of gravity of the object). There will be other specific properties according to the type such as the radius of a circle. In addition there might be general properties such as the area of the object, its distance from the origin and moment of inertia about it centre.

There are a number of ways in which such a hierarchy might be structured. We might have a package declaring a root abstract type and then another package with several derived types.

```
package Root is
  type Object is abstract tagged
    record
      X_Coord: Float;
      Y_Coord: Float;
    end record;

  function Area(O: Object) return Float is abstract;
  function MI(O: Object) return Float is abstract;
  function Distance(O: Object) return Float;
end Root;

package body Root is
  function Distance(O: Object) return Float is
  begin
    return Sqrt(O.X_Coord**2 + O.Y_Coord**2);
  end Distance;
end Root;
```

This package declares the root type and two abstract operations Area and MI (moment of inertia) and a concrete operation Distance. We might then have

```
with Root;
package Shapes is
  type Circle is new Root.Object with
    record
      Radius: Float;
    end record;

  function Area(C: Circle) return Float;
  function MI(C: Circle) return Float;

  type Triangle is new Root.Object with
    record
      A, B, C: Float;       -- lengths of sides
    end record;

  function Area(T: Triangle) return Float;
  function MI(T: Triangle) return Float;

-- and so on for other types such as Square

end Shapes;
```

(In the following discussion we will assume that use clauses are not being used. This is quite realistic because many projects forbid use clauses.)

Having declared some objects such as A_Circle and A_Triangle we can then apply the operations Area, Distance, and MI. In Ada 95 we write

```
A := Shapes.Area(A_Circle);
D := Shapes.Distance(A_Triangle);
M := Shapes.MI(A_Square);
```

Observe that the operation Distance is inherited and so is implicitly declared in the package Shapes for all types even though there is no mention of it in the text of the package Shapes. However, if we were using Ada 2005 and the prefixed notation then we could simply write

```
A := A_Circle.Area;
D := A_Triangle.Distance;
M := A_Square.MI;
```

and there is no mention of the package Shapes at all.

A clever friend then points out that by its nature Distance is the same for all types so it would be safer to avoid the risk of it getting changed by making it class wide. So we change the declaration of Distance in the package Root thus

```
function Distance(O: Object'Class) return Float;
```

and recompile our program. But the Ada 95 version won't recompile. Why? Because class wide operations are not inherited. So there is only one function Distance and it is declared in the package Root. So all our calls of Distance have to be changed to

```
D := Root.Distance(A_Triangle);
```

However, if we had been using the prefixed notation then there would have been nothing to change.

Our manager might then read about the virtues of child packages and tell us to restructure the whole thing as follows

```
package Geometry is
  type Object is abstract ...

  ... -- functions Area, MI, Distance
end Geometry;

package Geometry.Circles is
  type Circle is new Object with
    record
      Radius: Float;
    end record;

  ... -- functions Area, MI
end Geometry.Circles;

package Geometry.Triangles is
  type Triangle is new Object with
    record
      A, B, C: Float;
    end record;

  ... -- functions Area, MI
end Geometry.Triangles;

-- and so on
```

This is of course a much more beautiful structure and avoids having to write Root.Object when doing the extensions. But, horrors, our assignments in Ada 95 now have to be changed to

```
A := Geometry.Circles.Area(A_Circle);
D := Geometry.Distance(A_Triangle);
M := Geometry.Squares.MI(A_Square);
```

But the lucky programmer using Ada 2005 can still write

```
A := A_Circle.Area;
D := A_Triangle.Distance;
M := A_Square.MI;
```

and have a refreshing coffee (or a relaxing martini) while we are toiling with the editor.

Some time later the program might be extended to accommodate triangles that are specialized to be equilateral. This might be done by

```
package Geometry.Triangles.Equilateral is
  type Equilateral_Triangle is new Triangle with private;
  ...
private
  ...
end;
```

This type of course inherits all the operations of the type Triangle. We might now realize that the object A_Triangle of type Triangle was equilateral anyway and so it would be better to change it to be of type Equilateral_Triangle. The lucky Ada 2005 programmer will only have to change the declaration of the object but the poor Ada 95 programmer will have to change the calls on all its primitive operations such as

```
A := Geometry.Triangles.Area(A_Triangle);
```

to the corresponding

```
A := Geometry.Triangles.Equilateral.Area(A_Triangle);
```

Other advantages of the prefixed notation were mentioned in the Introduction. One is that it unifies the notation for calling a function with a single parameter and directly reading a component of the object. Thus we can write uniformly

```
X := A_Circle.X_Coord;
A := A_Circle.Area;
```

Of course if we were foolish and had a *visible* component Area as well as a function Area then we could not call the function in this way.

But now suppose we decide to make the root type private so that the coordinates cannot be changed inadvertently. Moreover we decide to provide functions to read them. So we have

```
package Geometry is
  type Object is abstract tagged private;
  function Area(O: Object) return Float is abstract;
  function MI(O: Object) return Float is abstract;
  function Distance(O: Object'Class) return Float;
  function X_Coord(O: Object'Class) return Float;
  function Y_Coord(O: Object'Class) return Float;

private
  type Object is tagged
    record
      X_Coord: Float;
      Y_Coord: Float;
    end record;

end Geometry;
```

Using Ada 95 we would now have to change statements such as

```
X := A_Triangle.X_Coord;
Y := A_Triangle.Y_Coord;
```

into

```
X := Geometry.X_Coord(A_Triangle);
Y := Geometry.Y_Coord(A_Triangle);
```

or (if we had not been wise enough to make the functions class wide) perhaps even

```
X := Geometry.Triangles.Equilateral.X_Coord(A_Triangle);
Y := Geometry.Triangles.Equilateral.Y_Coord(A_Triangle);
```

whereas in Ada 2005 we do not have to make any changes at all.

Another advantage mentioned in the Introduction is that when using access types explicit dereferencing is not necessary. Suppose we have

```
type Pointer is access all Geometry.Object'Class;
...
This_One: Pointer := A_Circle'Access;
```

In Ada 95 (assuming that X_Coord is a visible component) we have to write

```
Put(This_One.X_Coord); ...
Put(This_One.Y_Coord); ...
Put(Geometry.Area(This_One.all));
```

whereas in Ada 2005 we can uniformly write

```
Put(This_One.X_Coord); ...
Put(This_One.Y_Coord); ...
Put(This_One.Area);
```

and of course this remains unchanged if we make the coordinates into functions whereas the Ada 95 statements will need to be changed.

There are other structural changes that can occur during program development which are much easier to cope with using the prefix notation. For example, a class wide operation might be moved. And in the case of multiple interfaces to be described in the next section an operation might be moved from one interface to another.

It is clear that the prefixed notation has significant benefits both in terms of program clarity and for program maintenance.

Other variations on the rules for the use of the notation were considered. One was that the mechanism should apply to untagged types as well but this was rejected on the grounds that it might add to rather than reduce confusion in some cases. In any event, untagged types do not have class wide types so they are intrinsically simpler.

It is of course important to note that the first parameter of an operation plays a special role since in order to take advantage of the prefixed notation we have to ensure that the first parameter is a controlling parameter. Treating the first parameter specially can appear odd in some circumstances such as when there is symmetry among the parameters. Thus suppose we have a set package for creating and manipulating sets of integers

```
package Sets is
   type Set is tagged private;
   function Empty return Set;
   function Unit(N: Integer) return Set;
   function Union(S, T: Set) return Set;
   function Intersection(S, T: Set) return Set;
   function Size(S: Set) return Integer;
   ...
end Sets;
```

then we can apply the function Union in the traditional way

```
A, B, C: Set;
...
C := Sets.Union(A, B);
```

The object oriented addict can also write

```
C := A.Union(B);
```

but this destroys the obvious symmetry and is rather like sending 3 to be added to 2 mentioned at the beginning of this discussion.

Hopefully the mature programmer will use the OO notation wisely. Maybe its existence will encourage a more uniform style in which the first parameter is always a controlling operand wherever possible. Of course it cannot be used for functions which are tag indeterminate such as

```
function Empty return Set;
function Unit(N: Integer) return Set;
```

since there are no controlling parameters. If a subprogram has just one parameter (which is controlling) such as Size then the call just becomes X.Size and no parentheses are necessary.

Note that the prefix does not have to be simply the name of an object such as X, it could be a function call so we might write

```
N := Sets.Empty.Size;          -- N = 0
M := Sets.Unit(99).Size;       -- M = 1
```

with the obvious results as indicated.

4 Interfaces

In Ada 95, a derived type can really only have one immediate ancestor. This means that true multiple inheritance is not possible although curious techniques involving discriminants and generics can be used in some circumstances

General multiple inheritance has problems. Suppose that we have a type T with some components and operations. Perhaps

```
type T is tagged
   record
      A: Integer;
      B: Boolean;
   end record;

procedure Op1(X: T);
procedure Op2(X: T);
```

Now suppose we derive two new types from T thus

```
type T1 is new T with
   record
      C: Character;
   end record;

procedure Op3(X: T1);

-- Op1 and Op2 inherited, Op3 added

type T2 is new T with
   record
      C: Colour;
   end record;

procedure Op1(X: T2);
procedure Op4(X: T2);

-- Op1 overridden, Op2 inherited, Op4 added
```

Now suppose that we were able to derive a further type from both T1 and T2 by perhaps writing

```
type TT is new T1 and T2 with null record;      -- illegal
```

This is about the simplest example one could imagine. We have added no further components or operations. But what would TT have inherited from its two parents?

There is a general rule that a record cannot have two components with the same identifier so presumably it has just one component A and one component B. But what about C? Does it inherit the character or the colour? Or is it illegal because of the clash? Suppose T2 had a component D instead of C. Would that be OK? Would TT then have four components?

And then consider the operations. Presumably it has both Op1 and Op2. But which implementation of Op1? Is it the original Op1 inherited from T via T1 or the overridden version inherited from T2? Clearly it cannot have both. But there is no reason why it cannot have both Op3 and Op4, one inherited from each parent.

The problems arise when inheriting components from more than one parent and inheriting different *implementations* of the same operation from more than one parent. There is no problem with inheriting the same specification of an operation from two parents.

These observations provide the essence of the solution. At most one parent can have components and at most one parent can have concrete operations -- for simplicity we make them the same parent. But abstract operations can be inherited from several parents. This can be phrased as saying that this kind of multiple inheritance is about merging contracts to be satisfied rather than merging algorithms or state.

So Ada 2005 introduces the concept of an interface which is a tagged type with no components and no concrete operations. The idea of a null procedure as an operation of a tagged type is also introduced; this has no body but behaves as if it has a null body. Interfaces are only permitted to have abstract subprograms and null procedures as operations.

We will outline the ways in which interfaces can be declared and composed in a symbolic way and then conclude with a more practical example.

We might declare a package Pi1 containing an interface Int1 thus

```
package Pi1 is
   type Int1 is interface;
   procedure Op1(X: Int1) is abstract;
   procedure N1(X: Int1) is null;
end Pi1;
```

Note the syntax. It uses the new reserved word **interface**. It does not say **tagged** although all interface types are tagged. The abstract procedure Op1 has to be explicitly stated to be abstract as usual. The null procedure N1 uses new syntax as well. Remember that a null procedure behaves as if its body comprises a single null statement; but it doesn't actually have a concrete body.

The main type derivation rule then becomes that a tagged type can be derived from zero or one conventional tagged types plus zero or more interface types. Thus

```
type NT is new T and Int1 and Int2 with ... ;
```

where Int1 and Int2 are interface types. The normal tagged type if any has to be given first in the declaration. The first type is known as the parent so the parent could be a normal tagged type or an interface. The other types are known as progenitors. Additional components and operations are allowed in the usual way.

(The term progenitors may seem strange but the term ancestors in this context was confusing and so a new term was necessary. Progenitors comes from the Latin progignere, to beget, and so is very appropriate.)

It might have been thought that it would be quite feasible to avoid the formal introduction of the concept of an interface by simply saying that multiple parents are allowed provided only the first has components and concrete operations. However, there would have been implementation complexities with the risk of violating privacy and distributed overheads. Moreover, it would have caused maintenance problems since simply adding a component to a type or making one of its abstract operations concrete would cause errors elsewhere in the system if it was being used as a secondary parent. It is thus much better to treat interfaces as a fundamentally new concept. Another advantage is that this provides a new class of generic parameter rather neatly without complex rules for instantiations.

If the normal tagged type T is in a package Pt with operations Opt1, Opt2 and so on we could now write

```
with Pi1, Pt;
package PNT is
   type NT is new Pt.T and Pi1.Int1 with ... ;
   procedure Op1(X: NT);
   -- possibly other ops of NT
end PNT;
```

We must of course provide a concrete procedure for Op1 inherited from the interface Int1 since we have declared NT as a concrete type. We could also provide an overriding for N1 but if we do not then we simply inherit the null procedure of Int1. We could also override the inherited operations Opt1 and Opt2 from T in the usual way.

Interfaces can be composed from other interfaces thus

```
type Int2 is interface;
...
type Int3 is interface and Int1;
...
type Int4 is interface and Int1 and Int2;
...
```

Note the syntax. A tagged type declaration always has just one of **interface**, **tagged** and **with** (it doesn't have any if it is not a tagged type). When we derive interfaces in this way we can add new operations so that the new interface such as Int4 will have all the operations of both Int1 and Int2 plus possibly some others declared specifically as operations of Int4. All these operations must be abstract or null and there are fairly obvious rules regarding what happens if two or more of the ancestor interfaces have the same operation. Thus a null procedure overrides an abstract one but otherwise repeated operations must have the same profile.

Class wide types also apply to interface types. The class wide type Int1'Class covers all the types derived from the interface Int1 (both other interfaces as well as normal tagged types). We can then dispatch using an object of a concrete tagged type in that class in the usual way since we know that any abstract operation of Int1 will have been overridden. So we might have

```
type Int1_Ref is access all Int1'Class;
NT_Var: aliased NT;
Ref: Int1_Ref := NT_Var'Access;
```

Observe that conversion is permitted between the access to class wide type Int1_Ref and any access type that designates a type derived from the interface type Int1. We informally speak of a specific tagged type as implementing an interface from which it is derived (directly or indirectly). The phrase "implementing an interface" is not used

formally in the definition of Ada 2005 but it is useful for purposes of discussion.

Interfaces can also be used in private extensions and as generic parameters.

Thus

```
   type PT is new T and Int2 and Int3 with private;
   ...
private
   type PT is new T and Int2 and Int3 with null record;
```

An important rule regarding private extensions is that the full view and the partial view must agree with respect to the set of interfaces they implement. Thus although the parent in the full view need not be T but can be any type derived from T, the same is not true of the interfaces which must be such that they both implement the same set exactly. This rule is important in order to prevent a client type from overriding private operations of the parent if the client implements an interface added in the private part.

Generic parameters take the form

```
generic
   type FI is interface and Int1 and Int2;
package ...
```

and then the actual parameter must be an interface which implements all the ancestors Int1, Int2 etc. The formal could also just be **type** FI **is interface**; in which case the actual parameter can be any interface. There might be subprograms passed as further parameters which would require that the actual has certain operations. The interfaces Int1 and Int2 might themselves be formal parameters occurring earlier in the parameter list.

Interfaces (and formal interfaces) can also be limited thus

```
type LI is limited interface;
```

We can compose mixtures of limited and nonlimited interfaces but if any one of them is nonlimited then the resulting interface must not be specified as limited. This is because it must implement the equality and assignment operations implied by the nonlimited interface. Similar rules apply to types which implement one or more interfaces.

There are other forms of interfaces, namely synchronized interfaces, task interfaces, and protected interfaces. These bring support for polymorphic, class wide object oriented programming to the real time programming arena. They will be described in a later paper.

Having described the general ideas in somewhat symbolic terms, we will now discuss a more concrete example.

Before doing so it is important to emphasize that interfaces cannot have components and therefore if we are to perform multiple inheritance then we should think in terms of abstract operations to read and write components rather than the components themselves. This is standard OO thinking anyway because it preserves abstraction by hiding implementation details.

Thus rather than having a component such as Comp it is better to have a pair of operations. The function to read the component can simply be called Comp. A procedure to update the component might be Set_Comp. We will generally use this convention although it is not always appropriate to treat the components as unrelated entities.

Suppose now that we want to print images of the geometrical objects. We will assume that the root type is declared as

```
package Geometry is
   type Object is abstract tagged private;
   procedure Move(O: in out Object'Class; X, Y: Float);
   ...
private
   type Object is abstract tagged
      record
         X_Coord: Float := 0.0;
         Y_Coord: Float := 0.0;
      end record;
   ...
end;
```

The type Object is private and by default both coordinates have the value of zero. The procedure Move, which is class wide, enables any object to be moved to the location specified by the parameters.

Suppose also that we have a line drawing package with the following specification

```
package Line_Draw is
   type Printable is interface;
   type Colour is ... ;
   type Points is ... ;
   procedure Set_Hue(P: in out Printable; C: in Colour)
                                                  is abstract;
   function Hue(P: Printable) return Colour is abstract;
   procedure Set_Width(P: in out Printable; W: in Points)
                                                  is abstract;
   function Width(P: Printable) return Points is abstract;

   type Line is ... ;
   type Line_Set is ... ;

   function To_Lines(P: Printable) return Line_Set
                                                  is abstract;

   procedure Print(P: in Printable'Class);

private
   procedure Draw_It(L: Line; C: Colour; W: Points);

end Line_Draw;
```

The idea of this package is that it enables the drawing of an image as a set of lines. The attributes of the image are the hue and the width of the lines and there are pairs of subprograms to set and read these properties of any object of the interface Printable and its descendants. These operations are of course abstract.

In order to prepare an object in a form that can be printed it has to be converted to a set of lines. The function To_Lines converts an object of the type Printable into a set of lines;

again it is abstract. The details of various types such as Line and Line_Set are not shown.

Finally the package Line_Draw declares a concrete procedure Print which takes an object of type Printable'Class and does the actual drawing using the slave procedure Draw_It declared in the private part. Note that Print is class wide and is concrete. This is an important point. Although all primitive operations of an interface must be abstract this does not apply to class wide operations since these are not primitive.

The body of the procedure Print could take the form

```
procedure Print(P: in Printable'Class) is
  L: Line_Set := To_Lines(P);
  A_Line: Line;
begin
  loop
    -- iterate over the Line_Set and extract each line
    A_Line := ...
    Draw_It(A_Line, Hue(P), Width(P);
  end loop;
end Print;
```

but this is all hidden from the user. Note that the procedure Draw_It is declared in the private part since it need not be visible to the user.

One reason why the user has to provide To_Lines is that only the user knows about the details of how best to represent the object. For example the poor circle will have to be represented crudely as a polygon of many sides, perhaps a hectogon of 100 sides.

We can now take at least two different approaches. We can for example write

```
with Geometry, Line_Draw;
package Printable_Geometry is
  type Printable_Object is abstract new Geometry.Object
                    and Line_Draw.Printable with private;
  procedure Set_Hue(P: in out Printable_Object;
                                          C: in Colour);
  function Hue(P: Printable_Object) return Colour;
  procedure Set_Width(P: in out Printable_Object;
                                          W: in Points);
  function Width(P: Printable_Object) return Points;
  function To_Lines(P: Printable_Object) return Line_Set
                                          is abstract;

private
  ...
end Printable_Geometry;
```

The type Printable_Object is a descendant of both Object and Printable and all concrete types descended from Printable_Object will therefore have all the operations of both Object and Printable. Note carefully that we have to put Object first in the declaration of Printable_Object and that the following would be illegal

```
type Printable_Object is abstract new Line_Draw.Printable
        and Geometry.Object with private; --illegal
```

This is because of the rule that only the first type in the list can be a normal tagged type; any others must be interfaces. Remember that the first type is always known as the parent type and so the parent type in this case is Object.

The type Printable_Object is declared as abstract because we do not want to implement To_Lines at this stage. Nevertheless we can provide concrete subprograms for all the other operations of the interface Printable. We have given the type a private extension and so in the private part of its containing package we might have

```
private
  type Printable_Object is abstract new Geometry.Object
                        and Line_Draw.Printable with
    record
      Hue: Colour := Black;
      Width: Points := 1;
    end record;
end Printable_Geometry;
```

Just for way of illustration, the components have been given default values. In the package body the operations such as the function Hue are simply

```
function Hue(P: Printable_Object) return Colour is
begin
  return P.Hue;
end;
```

Luckily the visibility rules are such that this does not do an infinite recursion!

Note that the information containing the style components is in the record structure following the geometrical properties. This is a simple linear structure since interfaces cannot add components. However, since the type Printable_Object has all the operations of both an Object and a Printable, this adds a small amount of complexity to the arrangement of dispatch tables. But this detail is hidden from the user.

The key point is that we can now pass any object of the type Printable_Object or its descendants to the procedure

```
procedure Print(P: in Printable'Class);
```

and then (as outlined above) within Print we can find the colour to be used by calling the function Hue and the line width to use by calling the function Width and we can convert the object into a set of lines by calling the function To_Lines.

And now we can declare the various types Circle, Triangle, Square and so on by making them descendants of the type Printable_Object and in each case we have to implement the function To_Lines.

The unfortunate aspect of this approach is that we have to move the geometry hierarchy. For example the triangle package might now be

```
package Printable_Geometry.Triangles is
  type Printable_Triangle is new Printable_Object with
    record
      A, B, C: Float;
```

```
      end record;
   ... -- functions Area, To_Lines etc
end;
```

We can now declare a Printable_Triangle thus

```
A_Triangle: Printable_Triangle :=
    (Printable_Object with A => 4.0, B => 4.0, C => 4.0);
```

This declares an equilateral triangle with sides of length 4.0. Its private Hue and Width components are set by default. Its coordinates which are also private are by default set to zero so that it is located at the origin. (The reader can improve the example by making the components A, B and C private as well.)

We can conveniently move it to wherever we want by using the procedure Move which being class wide applies to all types derived from Object. So we can write

```
A_Triangle.Move(1.0, 2.0);
```

And now we can make a red sign

```
Sign: Printable_Triangle := A_Triangle;
```

Having declared the object Sign, we can give it width and hue and print it

```
Sign.Set_Hue(Red);
Sign.Set_Width(3);
Sign.Print;                    -- print thick red triangle
```

As we observed earlier this approach has the disadvantage that we had to move the geometry hierarchy. A different approach which avoids this is to declare printable objects of just the kinds we want as and when we want them.

So assume now that we have the package Line_Draw as before and the original package Geometry and its child packages. Suppose we want to make printable triangles and circles. We could write

```
with Geometry, Line_Draw;  use Geometry;
package Printable_Objects is
  type Printable_Triangle is new Triangles.Triangle and
                        Line_Draw.Printable with private;
  type Printable_Circle is new Circles.Circle and
                        Line_Draw.Printable with private;
  procedure Set_Hue(P: in out Printable_Triangle;
                                          C: in Colour);
  function Hue(P: Printable_Triangle return Colour;
  procedure Set_Width(P: in out Printable_Triangle;
                                          W: in Points);
  function Width(P: Printable_Triangle) return Points;
  function To_Lines(T: Printable_Triangle)
                                      return Line_Set;
  procedure Set_Hue(P: in out Printable_Circle;
                                          C: in Colour);
  function Hue(P: Printable_Circle) return Colour;
  procedure Set_Width(P: in out Printable_Circle;
                                          W: in Points);
  function Width(P: Printable_Circle) return Points;
  function To_Lines(C: Printable_Circle)
                                      return Line_Set;
private
  type Printable_Triangle is new Triangles.Triangle and
                        Line_Draw.Printable with
    record
      Hue: Colour := Black;
      Width: Points := 1;
    end record;

  type Printable_Circle is new Circles.Circle and
                        Line_Draw.Printable with
    record
      Hue: Colour := Black;
      Width: Points := 1;
    end record;

end Printable_Objects;
```

and the body of the package will provide the various subprogram bodies.

Now suppose we already have a normal triangle thus

```
A_Triangle: Geometry.Triangles.Triangle := ... ;
```

In order to print A_Triangle we first have to declare a printable triangle thus

```
Sign: Printable_Triangle;
```

and now we can set the triangle components of it using a view conversion thus

```
Triangle(Sign) := A_Triangle;
```

And then as before we write

```
Sign.Set_Hue(Red);
Sign.Set_Width(3);
Sign.Print_It;                -- print thick red triangle
```

This second approach is probably better since it does not require changing the geometry hierarchy. The downside is that we have to declare the boring hue and width subprograms repeatedly. We can make this much easier by declaring a generic package thus

```
with Line_Draw;  use Line_Draw;
generic
  type T is abstract tagged private;
package Make_Printable is
  type Printable_T is
          abstract new T and Printable with private;

  procedure Set_Hue(P: in out Printable_T;
                                          C: in Colour);
  function Hue(P: Printable_T) return Colour;
  procedure Set_Width(P: in out Printable_T;
                                          W: in Points);
  function Width(P: Printable_T) return Points;

private
  type Printable_T is abstract new T and Printable with
    record
      Hue: Colour := Black;
      Width: Points := 1;
    end record;
end;
```

This generic can be used to make any type printable. We simply write

```
package P_Triangle is new Make_Printable(Triangle);
type Printable_Triangle is new P_Triangle.Printable_T
                                        with null record;
function To_Lines(T: Printable_Triangle)
                                        return Line_Set;
```

The instantiation of the package creates a type Printable_T which has all the hue and width operations and the required additional components. However, it simply inherits the abstract function To_Lines and so itself has to be an abstract type. Note that the function To_Lines has to be especially coded for each type anyway unlike the hue and width operations which can be the same.

We now do a further derivation largely in order to give the type Printable_T the required name Printable_Triangle and at this stage we provide the concrete function To_Lines.

We can then proceed as before. Thus the generic makes the whole process very easy -- any type can be made printable by just writing three lines plus the body of the function To_Lines.

Hopefully this example has illustrated a number of important points about the use of interfaces. The key thing perhaps is that we can use the procedure Print to print anything that implements the interface Printable.

Earlier we stated that it was a common convention to provide pairs of operations to read and update properties such as Hue and Set_Hue and Width and Set_Width. This is not always appropriate. Thus if we have related components such as X_Coord and Y_Coord then although individual functions to read them might be sensible, it is undoubtedly better to update the two values together with a single procedure such as the procedure Move declared earlier. Thus if we wish to move an object from the origin (0.0, 0.0) to say (3.0, 4.0) and do it by two calls

```
Obj.Set_X_Coord(3.0);          -- first change X
Obj.Set_Y_Coord(4.0);          -- then change Y
```

then it seems as if it was transitorily at the point (3.0, 0.0). There are various other risks as well. We might forget to set one component or accidentally set the same component twice. And there could be big problems in a multitasking program.

Finally, as discussed earlier, null procedures are a new kind of subprogram and the user-defined operations of an interface must be null procedures or abstract subprograms − there is of course no such thing as a null function.

(Nonlimited interfaces do have one concrete operation and that is predefined equality; it could even be overridden with an abstract one.)

Null procedures will be found useful for interfaces but are in fact applicable to any types. As an example the package Ada.Finalization now uses null procedures for Initialize, Adjust, and Finalize as described in the Introduction.

5 Nested type extension

In Ada 95 type extension of tagged types has to be at the same level as the parent type. This can be quite a problem. In particular it means that all controlled types must be declared at library level because the root types Controlled and Limited_Controlled are declared in the library level package Ada.Finalization. The same applies to storage pools and streams because again the root types Root_Storage_Pool and Root_Stream_Type are declared in library packages.

This has a cumulative effect since if we write a generic unit using any of these types then that package can itself only be instantiated at library level. This enforces a very flat level of programming and hinders abstraction.

The problems can actually be illustrated without having to use controlled types or generics. As a simple example consider the following which is adapted from a text book [3]. It manipulates lists of colours and we assume that the type Colour is declared somewhere.

```
package Lists is
  type List is limited private;
  type Iterator is abstract tagged null record;
  procedure Iterate(IC: in Iterator'Class; L: in List);
  procedure Action(It: in out Iterator; C: in out Colour)
                                        is abstract;
private
  ...
end;
```

The idea is that a call of Iterate calls Action (by dispatching) on each object of the list and thereby gives access to the colour of that object. The user has to declare an extension of Iterator and a specific procedure Action to do whatever is required on each object.

Some readers may find this sort of topic confusing. It might be easier to understand if we look at the private part and body of the package Lists which might be

```
private
  type Cell is
    record
      Next: access Cell;          -- anonymous type
      C: Colour;
    end record;

  type List is access Cell;
end;

package body Lists is
  procedure Iterate(IC: in Iterator'Class; L: in List) is
    This: access Cell := L;
  begin
    while This /= null loop
      Action(IC, This.C);         -- dispatching call
                                  -- or IC.Action(This.C);
      This := This.Next;
    end loop;
  end Iterate;
end Lists;
```

Note the use of the anonymous access types which avoid the need to have an incomplete declaration of Cell in the private part and an explicit type conversion in the procedure Iterate.

Now suppose we wish to change the colour of every green object to red. We write (in some library level package)

```
type GTR_It is new Iterator with null record;

procedure Action(It: in out GTR_It; C: in out Colour) is
begin
  if C = Green then C := Red; end if;
end Action;

procedure Green_To_Red(L: in List) is
  It: GTR_It;
begin
  Iterate(It, L);            -- or It.Iterate(L);
end Green_To_Red;
```

This works but is not ideal. The type GTR_It and the procedure Action should not be declared outside the procedure Green_To_Red since they are really only part of its internal workings. But we cannot declare the type GTR_It inside the procedure in Ada 95 because that would be an extension at an inner level.

The extra facilities of the predefined library in Ada 2005 and especially the introduction of containers which are naturally implemented as generic units forced a reconsideration of the reasons for restricting type extension in Ada 95. The danger of nested extension of course is that values of objects could violate the accessibility rules and outlive their type declaration. It was concluded that type extension could be permitted at nested levels with the addition of just a few checks to ensure that the accessibility rules were not violated.

So in Ada 2005 the procedure Green_To_Red can be written as

```
procedure Green_To_Red(L: in List) is
  type GTR_It is new Iterator with null record;

  procedure Action(It: in out GTR_It; C: in out Colour) is
  begin
    if C = Green then C := Red; end if;
  end Action;

  It: GTR_It;
begin
  Iterate(It, L);            -- or It.Iterate(L);
end Green_To_Red;
```

and all the workings are now wrapped up within the procedure as they should be.

Note incidentally that we can use the notation It.Iterate(L); even though the type GTR_It is not declared in a package in this case. Remember that although we cannot add new dispatching operations to a type unless it is declared in a package specification, nevertheless we can always override existing ones such as Action.

This example is all quite harmless and nothing can go wrong despite the fact that we have performed the extension at an inner level. This is because the value It does not outlive the execution of the procedure Action.

But suppose we have a class wide object Global_It as in the following

```
with Lists; use Lists;
package body P is

  function Dodgy return Iterator'Class is
    type Bad_It is new Iterator with null record;

    procedure Action(It: in out Bad_It; C: in out Colour) is
    begin
      ...
    end Action;

    It: Bad_It;
  begin
    return It;
  end Dodgy;

  Global_It: Iterator'Class := Dodgy;
begin
  Global_It.Action(Red_For_Danger);      -- dispatches
end P;
```

Now we are in deep trouble. We have returned a value of the local type Bad_It, assigned it as the initial value to Global_It and then dispatched on it to the procedure Action. But the procedure Action that will be called is the one inside Dodgy and this does not exist anymore since we have left the function Dodgy. So this must not be allowed to happen.

So various accessibility checks are required. There is a check on the return from a function with a class wide result that the value being returned does not have the tag of a type at a deeper level than that of the function itself. So in this example there is a check on the return from the function Dodgy; this fails and raises Program_Error so all is well.

There are similar checks on class wide allocators and when using T'Class'Input or T'Class'Output. Some of these can be carried out at compile time but others have to be checked at run time and they also raise Program_Error if they fail.

Moreover, in order to implement the checks associated with T'Class'Input and T'Class'Output two additional functions are declared in the package Ada.Tags; these are

```
function Descendant_Tag(External: String;
                        Ancestor_Tag) return Tag;

function Is_Descendant_At_Same_Level
           (Descendant, Ancestor: Tag) return Boolean;
```

The use of these will be outlined in the next section.

6 Object factory functions

The Ada 95 Rationale (Section 4.4.1) [2] says "We also note that object oriented programming requires thought especially if variant programming is to be avoided. There is a general difficulty in finding out what is coming which is

particularly obvious with input-output; it is easy to write dispatching output operations but generally impossible for input." In this context, variant programming means messing about with case statements and so on.

The point about input-output is that it is easy to write a heterogeneous file but not so easy to read it. In the simple case of a text file we can just do a series of calls of Put thus

```
Put ("John is ");  Put(21, 0);  Put(" years old.");
```

But text input is not so easy unless we know the order of the items in the file. If we don't know the order then we really have to read the wretched thing a line at a time and then analyse the lines.

Ada 95 includes a mechanism for doing this relatively easily in the case of tagged types and stream input-output. Suppose we have a class of tagged types rooted at Root with various derived specific types T1, T2 and so on. We can then output a sequence of values X1, X2, X3 of a variety of these types to a file identified by the stream access value S by writing

```
Root'Class'Output(S, X1);
Root'Class'Output(S, X2);
Root'Class'Output(S, X3);
...
```

The various calls first write the tag of the specific type and then the value of the type. The tag corresponding to the type T1 is the string External_Tag(T1'Tag). Remember that External_Tag is a function in the predefined package Ada.Tags.

On input we can reverse the process by writing something like

```
declare
   X: Root'Class := Root'Class'Input(S);
begin
   Process(X);              -- now process the object in X
```

The call of Root'Class'Input first reads the external tag and then dispatches to the appropriate function Tn'Input according to the value of the tag. The function reads the value and this is now assigned as the initial value to the class wide variable X. We can then do whatever we want with X by perhaps dispatching to a procedure Process which deals with it according to its specific type.

This works in Ada 95 but it is all magic and done by smoke and mirrors inside the implementation. The underlying techniques are unfortunately not available to the user.

This means that if we want to devise our own stream protocol or maybe just process some values in circumstances where we cannot directly use dispatching then we have to do it all ourselves with if statements or case statements. Thus we might be given a tag value and separately some information from which we can create the values of the particular type. In Ada 95 we typically have to do something like

```
The_Tag: Ada.Tags.Tag;
A_T1: T1;                   -- series of objects of each
A_T2: T2;                   -- specific type
A_T3: T3;
...
The_Tag := Get_Tag( ... );          -- get the tag value
if The_Tag = T1'Tag then
   A_T1 := Get_T( ... );    -- get  value of specific type
   Process(A_T1);           -- process the object
elsif The_Tag = T2'Tag then
   A_T2 := Get_T( ... );    -- get  value of specific type
   Process(A_T2);           -- process the object
elsif
   ...
end if;
```

We assume that Get_T is a primitive function of the class rooted at Root. There is therefore a function for each specific type and the selection in the if statements is made at compile time by the normal overload rules. Similarly Process is also a primitive subprogram of the class of types.

This is all very tedious and needs careful maintenance if we add further types to the class.

Ada 2005 overcomes this problem by providing a generic object constructor function. Its specification is

```
generic
   type T (<>) is abstract tagged limited private;
   type Parameters (<>) is limited private;
   with function Constructor(Params: access Parameters)
                                          return T is abstract;
function Ada.Tags.Generic_Dispatching_Constructor
         (The_Tag: Tag; Params: access Parameters)
                                          return T'Class;
```

This generic function works for both limited and nonlimited types. Remember that a nonlimited type is allowed as an actual generic parameter corresponding to a limited formal generic type. The generic function Generic_Dispatching_Constructor is Pure and has convention Intrinsic.

Note carefully the formal function Constructor. This is an example of a new kind of formal generic parameter introduced in Ada 2005. The distinctive feature is the use of **is abstract** in its specification. The interpretation is that the actual function must be a dispatching operation of a tagged type uniquely identified by the profile of the formal function. The actual operation can be concrete or abstract. Remember that the overriding rules ensure that the specific operation for any concrete type will always have a concrete body. Note also that since the operation is abstract it can only be called through dispatching.

In this example, it therefore has to be a dispatching operation of the type T since that is the only tagged type involved in the profile of Constructor. We say that T is the controlling type. In the general case, the controlling type does not itself have to be a formal parameter of the generic unit but usually will be as here.

Formal abstract subprograms can of course be procedures as well as functions. It is important that there is exactly one

controlling type in the profile. Thus given that TT1 and TT2 are tagged types then the following would both be illegal

```
with procedure Do_This(X1: TT1; X2: TT2) is abstract;
                                                -- illegal
with function Fn(X: Float) return Float is abstract;
                                                -- illegal
```

The procedure Do_This is illegal because it has two controlling types TT1 and TT2. Remember that we can declare a subprogram with parameters of more than one tagged type but it can only be a dispatching operation of one tagged type. The function Fn is illegal because it doesn't have any controlling types at all (and so could never be called in a dispatching call anyway).

The formal function Constructor is legal because only T is tagged; the type Parameters which also occurs in its profile is not tagged.

And now to return to the dispatching constructor. The idea is that we instantiate the generic function with a (root) tagged type T, some type Parameters and the dispatching function Constructor. The type Parameters provides a means whereby auxiliary information can be passed to the function Constructor.

The generic function Generic_Dispatching_Constructor takes two parameters, one is the tag of the type of the object to be created and the other is the auxiliary information to be passed to the dispatching function Constructor.

Note that the type Parameters is used as an access parameter in both the generic function and the formal function Constructor. This is so that it can be matched by the profile of the attribute Input whose specification is

```
function T'Input(Stream: access Root_Stream_Type'Class)
                                                return T;
```

Suppose we instantiate Generic_Dispatching_Constructor to give a function Make_T. A call of Make_T takes a tag value, dispatches to the appropriate Constructor which creates a value of the specific tagged type corresponding to the tag and this is finally returned as the value of the class wide type T'Class as the result of Make_T. It's still magic but anyone can use the magic and not just the magician implementing stream input-output.

We can now do our abstract problem as follows

```
function Make_T is
      new Generic_Dispatching_Constructor
                                (Root, Params, Get_T);

...
declare
  Aux: Params := ... ;
  A_T: Root'Class:= Make_T(Get_Tag( ... ), Aux);
begin
  Process(A_T);          -- dispatch to process the object
end;
```

We no longer have the tedious sequence of if statements and the calls of Get_T and Process are dispatching calls.

The previously magic function T'Class'Input can now be implemented in a very natural way by something like

```
function Dispatching_Input is
      new Generic_Dispatching_Constructor
                (T, Root_Stream_Type'Class, T'Input);

function T_Class_Input
 (S: access Root_Stream_Type'Class) return T'Class is
      -- read tag as string from stream
  The_String: String := String'Input(S);
      -- convert to a tag
  The_Tag: Tag := Descendant_Tag(The_String, T'Tag);
 begin
  -- now dispatch to the appropriate function Input
  return Dispatching_Input(The_Tag, S);
end T_Class_Input;

for T'Class'Input use T_Class_Input;
```

The body could of course be written as one giant statement

```
return Dispatching_Input
            (Descendant_Tag(String'Input(S), T'Tag), S);
```

but breaking it down hopefully clarifies what is happening.

Note the use of Descendant_Tag rather than Internal_Tag. Descendant_Tag is one of a few new functions introduced into the package Ada.Tags in Ada 2005. Streams did not work very well for nested tagged types in Ada 95 because of the possibility of multiple elaboration of declarations (as a result of tasking and recursion); this meant that two descendant types could have the same external tag value and Internal_Tag could not distinguish them. This is not an important problem in Ada 95 as nested tagged types are rarely used. In Ada 2005 the situation is potentially made worse because of the possibility of nested type extension.

The goal in Ada 2005 is simply to ensure that streams do work with types declared at the same level and to prevent erroneous behaviour otherwise. The goal is not to permit streams to work with the nested extensions introduced in Ada 2005. Any attempt to do so will result in Tag_Error being raised.

Note that we cannot actually declare an attribute function such as T'Class'Input by directly using the attribute name. We have to use some other identifier such as T_Class_Input and then use an attribute definition clause as shown above.

Observe that T'Class'Output can be implemented as

```
procedure T_Class_Output
   (S: access Root_Stream_Type'Class; X: in T'Class) is
begin
  if not Is_Descendant_At_Same_Level(X'Tag, T'Tag) then
    raise Tag_Error;
  end if;
  String'Output(S, External_Tag(X'Tag));
  T'Output(S, X);
end T_Class_Output;

for T'Class'Output use T_Class_Output;
```

Remember that streams are designed to work only with types declared at the same accessibility level as the parent type T. The call of Is_Descendant_At_Same_Level, which is another new function in Ada 2005, ensures this.

We can use the generic constructor to create our own stream protocol. We could in fact replace T'Class'Input and T'Class'Output or just create our own distinct subsystem. One reason why we might want to use a different protocol is when the external protocol is already given such as in the case of XML.

Note that it will sometimes be the case that there is no need to pass any auxiliary parameters to the constructor function in which case we can declare

```
type Params is null record;
Aux: aliased Params := (null record);
```

Another example can be based on part of the program Magic Moments in [3]. This reads in the values necessary to create various geometrical objects such as a Circle, Triangle, or Square which are derived from an abstract type Object. The values are preceded by a letter C, T or S as appropriate. The essence of the code is

```
Get(Code_Letter);
case Code_Letter is
  when 'C' => Object_Ptr := Get_Circle;
  when 'T' => Object_Ptr := Get_Triangle;
  when 'S' => Object_Ptr := Get_Square;
  ...
end case;
```

The types Circle, Triangle, and Square are derived from the root type Object and Object_Ptr is of the type **access** Object'Class. The function Get_Circle reads the value of the radius from the keyboard, the function Get_Triangle reads the values of the lengths of the three sides from the keyboard and so on.

The first thing to do is to change the various constructor functions such as Get_Circle into various specific overridings of a primitive operation Get_Object so that we can dispatch on it.

Rather than just read the code letter we could make the user type the external tag string and then we might have

```
function Make_Object is
      new Generic_Dispatching_Constructor
                        (Object, Params, Get_Object);

...
S: String := Get_String;
...
Object_Ptr := new Object'
                    (Make_Object(Internal_Tag(S), Aux));
```

but this is very tedious because the user now has to type the external tag which will be an implementation defined mess of characters. Observe that the string produced by a call of Expanded_Name such as

```
OBJECTS.CIRCLE
```

cannot be used because it will not in general be unique and so there is no reverse function. (It is not generally unique because of tasking and recursion.) But Expanded_Name is useful for debugging purposes.

In these circumstances the best way to proceed is to invent some sort of registration system to make a map to convert the simple code letters into the tag. We might have a package

```
with Ada.Tags; use Ada.Tags;
package Tag_Registration is
  procedure Register(The_Tag: Tag; Code: Character);
  function Decode(Code: Character) return Tag;
end;
```

and then we can write

```
Register(Circle'Tag, 'C');
Register(Triangle'Tag, 'T');
Register(Square'Tag, 'S');
```

And now the program to read the code and then make the object becomes simply

```
Get(Code_Letter);
Object_Ptr := new Object'
            (Make_Object(Decode(Code_Letter), Aux));
```

and there are no case statements to maintain.

The really important point about this example is that if we decide at a later date to add more types such as 'P' for Pentagon and 'H' for Hexagon then all we have to do is register the new code letters thus

```
Register(Pentagon'Tag, 'P');
Register(Hexagon'Tag, 'H');
```

and nothing else needs changing. This registration can conveniently be done when the types are declared.

The package Tag_Registration could be implemented trivially as follows by

```
package body Tag_Registration is
  Table: array (Character range 'A' .. 'Z') of Tag :=
                                  (others => No_Tag);
  procedure Register(The_Tag: Tag; Code: Character) is
  begin
    Table(Code) := The_Tag;
  end Register;

  function Decode(Code: Character) return Tag is
  begin
    return Table(Code);
  end Decode;
end Tag_Registration;
```

The constant No_Tag is a value of the type Tag which does not represent an actual tag. If we forget to register a type then No_Tag will be returned by Decode and this will cause Make_Object to raise Tag_Error.

A more elegant registration system could be easily implemented using the container library which will be described in a later paper.

Note that any instance of Generic_Dispatching_Constructor checks that the tag passed as parameter is indeed that of a type descended from the root type T and raises Tag_Error if it is not.

In simple cases we could in fact perform that check for ourselves by writing something like

```
   Trial_Tag: Tag := The_Tag;
loop
   if Trial_Tag = T'Tag then exit; end if;
   Trial_Tag := Parent_Tag(Trial_Tag);
   if Trial_Tag = No_Tag then raise Tag_Error; end if;
end loop;
```

The function Parent_Tag and the constant No_Tag are further items in the package Ada.Tags whose specification in Ada 2005 is

```
package Ada.Tags is
   type Tag is private;
   No_Tag: constant Tag;

   function Expanded_Name(T: Tag) return String;
   ...   -- also Wide and Wide_Wide versions
   function External_Tag(T: Tag) return String;
   function Internal_Tag(External: String) return Tag;
   function Descendant_Tag(External: String;
                           Ancestor: Tag) return Tag;
   function Is_Descendant_At_Same_Level(Descendant,
                           Ancestor: Tag) return Boolean;
   function Parent_Tag(T: Tag) return Tag;
   type Tag_Array is (Positive range <>) of Tag;
   function Interface_Ancestor_Tags(T: Tag) return Tag;

   Tag_Error: exception;
private
   ...
end Ada.Tags;
```

The function Parent_Tag returns No_Tag if the parameter T of type Tag has no parent which will be the case if it is the ultimate root type of the class. As mentioned earlier, two other new functions Descendant_Tag and Is_Descendant_At_Same_Level are necessary to prevent the misuse of streams with types not all declared at the same level.

There is also a function Interface_Ancestor_Tags which returns the tags of all the ancestors as an array. This includes the tag of T itself, its parent, any progenitors and all their ancestors as well.

Finally note that the introduction of 16- and 32-bit characters in identifiers means that functions also have to be provided to return the images of identifiers as a Wide_String or Wide_Wide_String. So we have functions Wide_Expanded_Name and Wide_Wide_Expanded_Name.

7 Overriding and overloading

One of the key goals in the design of Ada was to encourage the writing of correct programs. It was intended that the structure, strong typing, and so on should ensure that many errors which are not detected by most languages until run time should be caught at compile time in Ada. Unfortunately the introduction of type extension and overriding in Ada 95 produced a situation where careless errors in subprogram profiles lead to errors which are awkward to detect.

The Introduction described two typical examples. The first concerns the procedure Finalize. Consider

```
with Ada.Finalization;  use Ada.Finalization;
package Root is
   type T is new Controlled with ... ;
   procedure Op(Obj: in out T; Data: in Integer);
   procedure Finalise(Obj: in out T);
end Root;
```

We have inadvertently written Finalise rather than Finalize. This means that Finalize does not get overridden as expected and so the expected behaviour does not occur on finalization of objects of type T.

In Ada 2005 we can prefix the declaration with **overriding**

```
   overriding
   procedure Finalize(Obj: in out T);
```

And now if we inadvertently write Finalise then this will be detected during compilation.

Similar errors can occur in a profile. If we write

```
package Root.Leaf is
   type NT is new T with null record;
   overriding                          -- overriding indicator
   procedure Op(Obj: in out NT; Data: in String);
end Root.Leaf;
```

then the compiler will detect that the new procedure Op has a parameter of type String rather than Integer.

However if we do want a new operation then we can write

```
   not overriding
   procedure Op(Obj: in out NT; Data: in String);
```

The overriding indicators can also be used with abstract subprograms, null procedures, renamings, instantiations, stubs, bodies and entries (we will deal with entries in the paper on tasking). So we can have

```
overriding
procedure Pap(X: TT) is abstract;

overriding
procedure Pep(X: TT) is null;

overriding
procedure Pip(Y: TT) renames Pop;

not overriding
procedure Poop is new Peep( ... );

overriding
procedure Pup(Z: TT) is separate;

overriding
procedure Pup(X: TT) is
begin ... end Pup;
```

We do not need to apply an overriding indicator to both a procedure specification and body but if we do then they naturally must not conflict. It is expected that overriding indicators will typically only be given on specifications but they would be appropriate in the case of a body standing alone as in the example of Action in the previous section. So we might have

```
procedure Green_To_Red(L: in List) is
  type GTR_It is new Iterator with null record;

  overriding
  procedure Action(It: in out GTR_It; C: in out Colour) is
  begin
    if C = Green then C := Red; end if;
  end Action;
...
```

The overriding indicators are optional for two reasons. One is simply for compatibility with Ada 95. The other concerns awkward problems with private types and generics.

Consider

```
package P is
  type NT is new T with private;
  procedure Op(X: T);
private
```

Now suppose the type T does not have an operation Op. Then clearly it would be wrong to write

```
package P is
  type NT is new T with private;          -- T has no Op
  overriding                              -- illegal
  procedure Op(X: T);
private
```

because that would violate the information known in the partial view.

But suppose that in fact it turns out that in the private part the type NT is actually derived from TT (itself derived from T) and that TT does have an operation Op.

```
private
  type NT is new TT with ...      -- TT has Op
end P;
```

In such a case it turns out in the end that Op is in fact overriding after all. We can then put an overriding indicator on the body of Op since at that point we do know that it is overriding.

Equally of course we should not specify **not overriding** for Op in the visible part because that might not be true either (since it might be that TT does have Op). However if we did put **not overriding** on the partial view then that would not in itself be an error but would simply constrain the full view not to be overriding and thus ensure that TT does not have Op.

Of course if T itself has Op then we could and indeed should put an overriding indicator in the visible part since we know that to be the truth at that point.

The general rule is not to lie. But the rules are slightly different for **overriding** and **not overriding**. For **overriding** it must not lie at the point concerned. For **not overriding** it must not lie anywhere.

This asymmetry is a bit like presuming the prisoner is innocent until proved guilty. We sometimes start with a view in which an operation appears not to be overriding and then later on we find that it is overriding after all. But the reverse never happens -- we never start with a view in which it is overriding and then later discover that it was not. So the asymmetry is real and justified.

There are other similar but more complex problems with private types concerning implicit declarations where the implicit declaration turns up much later and is overriding but has no physical presence on which to hang the indicator. It was concluded that by far the best approach to these problems was just to say that the overriding indicator is always optional. We cannot expect to find all the bugs in a program through syntax and static semantics; the key goal here is to provide a simple way of finding most of them.

Similar problems arise with generics. As is usual with generics the rules are checked in the generic itself and then rechecked upon instantiation (in this case for uses within both the visible part and private part of the specification). Consider

```
generic
  type GT is tagged private;
package GP is
  type NT is new GT with private;
  overriding                          -- illegal, GT has no Op
  procedure Op(X: NT);
private
```

This has to be illegal because GT has no operation Op. Of course the actual type at instantiation might have Op but the check has to pass both in the generic and in the instantiation.

On the other hand saying **not overriding** is allowed

```
generic
  type GT is tagged private;
package GP is
  type NT is new GT with private;
  not overriding                      -- legal, GT has no Op
  procedure Op(X: NT);
private
```

However, in this case we cannot instantiate GP with a type that does have an operation Op because it would fail when checked on the instantiation. So in a sense this imposes a further contract on the generic. If we do not want to impose this restriction then we must not give an overriding indicator on the procedure Op for NT.

Another situation arises when the generic formal is derived

```
generic
  type GT is new T with private;
package GP is
  type NT is new GT with private;
```

```
   overriding                    -- legal if T has Op
   procedure Op(X: NT);
private
```

In this case it might be that the type T does have an operation Op in which case we can give the overriding indicator.

We might also try

```
generic
   type GT is tagged private;
   with procedure Op(X: GT);
package GP is
   type NT is new GT with private;
   overriding              -- illegal, Op not primitive
   procedure Op(X: NT);
private
```

But this is incorrect because although GT has to have an operation corresponding to Op as specified in the formal parameter list, nevertheless it does not have to be a primitive operation nor does it have to be called Op and thus it isn't inherited.

It should also be observed that overriding indicators can be used with untagged types although they have been introduced primarily to avoid problems with dispatching operations. Consider

```
package P is
   type T is private;
   function "+" (Left, Right: T) return T;
private
   type T is range 0 .. 100;        -- "+" overrides
end P;
```

as opposed to

```
package P is
   type T is private;
   function "+" (Left, Right: T) return T;
private
   type T is new TT;                -- "+" does not override
end P;
```

The point is that the partial view does not reveal whether overriding occurs or not -- nor should it since either implementation ought to be acceptable. We should therefore remain silent regarding overriding in the partial view. This is similar to the private extension and generic cases discussed earlier. Inserting **overriding** would be illegal on both examples, while **not overriding** would be allowed only on the second one (which would constrain the implementation as in the previous examples). Again, it is permissible to put an overriding indicator on the body of "+" to indicate whether or not it does override.

It is also possible for a subprogram to be primitive for more than one type (this cannot happen for more than one tagged type but it can happen for untagged types or one tagged type and some untagged types). It could then be overriding for some types and not overriding for others. In such a case it is considered to be overriding as a whole and any indicator should reflect this.

The possibility of having a pragma which would enforce the use of overriding indicators (so that they too could not be inadvertently omitted) was eventually abandoned largely because of the private type and generic problem which made the topic very complicated.

Note the recommended layout, an overriding indicator should be placed on the line before the subprogram specification and aligned with it. This avoids disturbing the layout of the specification.

It is hoped that programmers will use overriding indicators freely. As mentioned in the Introduction, they are very valuable for preventing nasty errors during maintenance. Thus if we add a further parameter to an operation such as Op for a root type and all type extensions have overriding indicators then the compiler will report an error if we do not modify the operators of all the derived types correctly.

We now turn to a minor change in the overriding rules for functions with controlling results.

The reader may recall the general rule in Ada 95 that a function that is a primitive operation of a tagged type and returns a value of the type, must always be overridden when the type is extended. This is because the function for the extended type must create values for the additional components. This rule is sometimes phrased as saying that the function "goes abstract" and so has to be overridden if the extended type is concrete. The irritating thing about the rule in Ada 95 is that it applies even if there are no additional components.

Thus consider a generic version of the set package of Section 3

```
generic
   type Element is private;
package Sets is
   type Set is tagged private;
   function Empty return Set;
   function Unit(E: Element) return Set;
   function Union(S, T: Set) return Set;
   function Intersection(S, T: Set) return Set;
   ...
end Sets;
```

Now suppose we declare an instantiation thus

```
package My_Sets is new Sets(My_Type);
```

This results in the type Set and all its operations being declared inside the package My_Sets. However, for various reasons we might wish to have the type and its operations at the current scope. One reason could just be for simplicity of naming so that we do not have to write My_Sets.Set and My_Sets.Union and so on. (We might be in a regime where use clauses are forbidden.) An obvious approach is to derive our own type locally so that we have

```
package My_Sets is new Sets(My_Type);
type My_Set is new My_Sets.Set with null record;
```

Another situation where we might need to do this is where we wish to use the type Set as the full type for a private type thus

```
   type My_Set is private;
private
   package My_Sets is new Sets(My_Type);
   type My_Set is new My_Sets.Set with null record;
```

But this doesn't work nicely in Ada 95 since all the functions have controlling results and so "go abstract" and therefore have to be overridden with wrappers thus

```
function Union(S, T: My_Set) return My_Set is
begin
   return My_Set(My_Sets.Union(My_Sets.Set(S),
                               My_Sets.Set(T)));
end Union;
```

This is clearly a dreadful nuisance. Ada 2005 sensibly allows the functions to be inherited provided that the extension is visibly null (and that there is no new discriminant part) and so no overriding is required. This new facility will be much appreciated by users of the new container library in Ada 2005 which has just this style of generic packages which export tagged types.

The final topic to be discussed concerns a problem with overloading and untagged types. Remember that the concept of abstract subprograms was introduced into Ada 95 largely for the purpose of tagged types. However it can also be used with untagged types on derivation if we do not want an operation to be inherited. This often happens with types representing physical measurements. Consider

```
type Length is new Float;
type Area is new Float;
```

These types inherit various undesirable operations such as multiplying a length by a length to give a length when of course we want an area. We can overcome this by overriding them with abstract operations. Thus

```
function "*" (L, R: Length) return Length is abstract;
function "*" (L, R: Area) return Area is abstract;
function "*" (L, R: Length) return Area;
```

We have also declared a function to multiply two lengths to give an area. So now we have two functions multiplying two lengths, one returns a length but is abstract and so can never be called and the other correctly returns an area.

Now suppose we want to print out some values of these types. We might declare a couple of functions delivering a string image thus

```
function Image(L: Length) return String;
function Image(L: Area) return String;
```

And then we decide to write

```
X: Length := 2.5;
...
Put_Line(Image(X * X));          -- ambiguous in 95
```

This fails to compile in Ada 95 since it is ambiguous because both Image and "*" are overloaded. The problem is that although the function "*" returning a length is abstract it nevertheless is still there and is considered for overload resolution. So we don't know whether we are calling Image on a length or on an area because we don't know which "*" is involved.

So declaring the operation as abstract does not really get rid of the operation at all, it just prevents it from being called but its ghost lives on and is a nuisance.

In Ada 2005 this is overcome by a new rule that says "abstract nondispatching subprograms are ignored during overload resolution". So the abstract "*" is ignored and there is no ambiguity in Ada 2005.

Note that this rule does not apply to dispatching operations of tagged types since we might want to dispatch to a concrete operation of a descendant type. But it does apply to operations of a class-wide type.

References

[1] ISO/IEC JTC1/SC22/WG9 N412 (2002) *Instructions to the Ada Rapporteur Group from SC22/WG9 for Preparation of the Amendment*.

[2] *Ada 95 Rationale* (1995) LNCS 1247, Springer-Verlag.

[3] J. G. P. Barnes (1998) *Programming in Ada 95*, 2nd ed., Addison-Wesley.

Stop press: Since the publication of the Introduction part of the Rationale for Ada 2005 in the previous issue of Ada User Journal, some small changes have been made regarding a number of topics. The main change which affects the level of detail given in the Introduction is that a gratuitous all is no longer allowed with anonymous access types. This will be explained in detail in the instalment of the Rationale in the next issue of Ada User Journal. Order your copy now!!

Ada-Europe 2004 Sponsors

ACT Europe
Contact: Zépur Blot
8 Rue de Milan, F-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

Aonix
Contact: Anne Chapey
66/68, Avenue Pierre Brossolette, 92247 Malakoff, France
Tel: +33-1-41-48-10-10 Fax: +33-1-41-48-10-20
Email : info@aonix.fr URL : www.aonix.fr

Artisan Software Tools Ltd
Contact: Emma Allen
Suite 701, Eagle Tower, Montpellier Drive, Cheltenham, GL50 1TA, UK
Tel: +44-1242-229300 Fax: +44-1242-229301
Email : info.uk@artisansw.com URL : www.artisansw.com

Green Hills Software Ltd
Contact: Christopher Smith
Dolphin House, St Peter Street, Winchester, Hampshire, SO23 8BW, UK
Tel: +44-1962-829820 Fax: +44-1962-890300
Email : chriss@ghs.com URL : www.ghs.com

I-Logix
Contact: Martin Stacey
1 Cornbrash Park, Bumpers Way, Chippenham, Wiltshire, SN14 6RA, UK
Tel: +44-1249-467-600 Fax: +44-1249-467-610
Email : info_euro@ilogix.com URL : www.ilogix.com

LDRA Ltd
Contact: Jim Kelly
24 Newtown Road, Newbury, Berkshire, RG14 7BN, UK
Tel: +44-1635-528-828 Fax: +44-1635-528-657
Email: info@ldra.com URL: www.ldra.com

Praxis Critical Systems Ltd
Contact: Rod Chapman
20 Manvers Street, Bath, BA1 1PX, UK
Tel: +44-1225-823763 Fax: +44-1225-469006
Email : sparkinfo@praxis-cs.co.uk URL : www.sparkada.com

Scientific Toolworks Inc
Contact: Matthew Bergeson
321 N. Mall Drive Suite I-201, St. George, UT 84790, USA
Tel: +1-435-627-2529 Fax: +1-877-512-0765
Email: sales@scitools.com URL: www.scitools.com

TNI Europe Limited
Contact: Pam Flood
Triad House, Mountbatten Court, Worrall Street, Congleton, Cheshire CW12 1DT, UK
Tel: +44-1260-29-14-49 Fax: +44-1260-29-14-49
Email: info@tni-europe.com URL: www.tni-europe.com