Minutes of Electronic ARG Meeting 60C

11 February 2019

Attendees: Steve Baird, Randy Brukardt, Gary Dismukes, Bob Duff, Jeff Cousins, Brad Moore, Jean-Pierre Rosen, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega (left at 12:07).

Observers: Richard Wai.

Meeting Summary

The meeting convened on Monday, 11 February 2019 at 11:01 hours EST and adjourned at 14:03 hours EST. The meeting was held using Zoom. The meeting covered some of the AIs on the agenda.

AI Summary

The following AI was approved:

AI12-0236-1/08 declare expressions (10-1-0)

The following AIs were approved with editorial changes:

AI12-0262-1/09 Map-Reduce attributes (10-0-1)
AI12-0266-1/11 Parallel container iterators (8-0-2)
AI12-0306-1/01 Split null array aggregates from positional array aggregates (9-1-0)
AI12-0307-1/01 Resolution of aggregates (8-0-2)
AI12-0308-1/01 Declared entities need to be declarations (10-0-0)
AI12-0309-1/01 Missing checks for pragma Suppress (7-1-2)
AI12-0310-1/01 Specifying the private parts of a packages in aspect Global (10-0-0)

The intention of the following AI was approved but it requires a rewrite:

AI12-0311-1/01 Suppressing client-side assertions for language-defined units (9-0-1)

The following AIs were discussed and assigned to an editor:

AI12-0191-1/02 Clarify “part” for type invariants
AI12-0312-1/00 Examples for Ada 2020
AI12-0313-1/00 Introduction for Ada 2020

Detailed Minutes

Welcome

Steve announces “Welcome to today's ARG thingy”. He gets a laugh from the group.

Apologies

Erhard Ploedereder said that he would be in another meeting and unable to attend. Tullio Vardanega tell us that he has to leave for travel at 12:10.

Previous Meeting Minutes

No one has any changes or comments on the minutes. Approve minutes: 11-0-0.

Date and Venue of the Next Meeting

Randy proposes the next electronic meeting to be on March 11th at 11:00 EST. Steve and Gary will be in Paris that day, but they will have arrived long before the meeting time, so there shouldn't be a problem. We chose March 11th. [But see below.]

Our next face-to-face meeting is scheduled for June 14-16 in Warsaw Poland, immediately after the Ada-Europe conference.

We again discuss the dates for the fall meeting. AdaCore's HUB week is October 8-10, so many AdaCore employees will be on the east coast then. We ought to meet the weekend before or after. Bob notes that the weekend after is Columbus Day. That's not a big deal, but we might as well avoid it anyway. So we chose October 5-7, in Boston, with the meeting ending in time on the 7th for all of the AdaCore people to leave for New York.

Tucker asks if we need a meeting at all in the fall. Randy says that our primary work hopefully will be done then, but there could be fixes to be handled. Especially if ACATS and implementation work is ramping up (those have a way of digging out problems in the Standard). We also could do some ACATS design work if we don't have enough for the full time. In any case, we don't know now if there will be any major problems uncovered during reviews, so we should set aside the dates in order to be prepared. We can decide in June whether to have a fall meeting or not.

As we were nearing the end of the allotted time, Ed raised the issue that we are not progressing any of the relatively finished AIs, especially those on the “other” list. Steve suggests that we have an extra meeting in two weeks to hopefully make more of a dent in those AIs. Several people have conflicts with Monday, February 25th, so we settle on Tuesday, February 26th, 11:00 hours EST. We still will plan on a meeting on March 11th as well.

RM Review Plan

Randy notes that he didn't think there was enough time for a review after he completed RM work this time, it made more sense to wait until after this meeting when more AIs have been approved and added. He still has AI12-0020-1 to do, which will require a new command (thus some programming) in order to be able to move attribute definitions.

The review plan is unchanged. We will distribute a draft Standard early next week. That will contain all of the approved changes to date outside of some of the containers changes. He will assign people approximately equivalent sections of the Standard to review. Hopefully, these can be turned around in time for our March meeting, so we can begin to address the issues that inevitably will come up.

Unfinished Action Items

Steve Baird has the only unfinished action item, the “dynamic accessibility is a pain to implement” AI (AI12-0016-1). Justin is looking at implementing some solution for this problem in GNAT, so there may actually be something to write about this in the future.

Current Action Items

The combined unfinished old action items and new action items from the meeting are shown below.

Everyone:

- Review their assigned portion of the draft Standard, after it is posted by Randy.

Steve Baird:

- AI12-0016-1
- AI12-0191-1
- AI12-0311-1
- AI12-0313-1 (primary author, get help from Tucker)

Randy Brukardt:

- Post a draft Standard as soon as practical, and assign review sections.

Editorial changes only:

- AI12-0262-1
- AI12-0266-1
- AI12-0306-1

- AI12-0307-1
- AI12-0308-1
- AI12-0309-1
- AI12-0310-1

Jeff Cousins:

- AI12-0312-1 (Help Brad and Ed)

Brad Moore:

- AI12-0312-1 (primary author, get help from Ed and Jeff)

Ed Schonberg:

- AI12-0312-1 (Help Brad and Jeff)

Tucker Taft:

- AI12-0313-1 (Help Steve)
- Propose a term for new objects of a limited type (see discussion of AI12-0236-1), as well as uses of this term.

Detailed Review

The minutes cover detailed review of Ada 2012 AIs (AI12s). The AI12s are presented in numeric order, which is not necessarily the order in which they were discussed. Votes are recorded as “for”-“against”-“abstentions”. For instance, a vote of 6-1-2 would have had six votes for, one vote against, and two abstentions.

If a paragraph number is identified as coming from the working Ada 202x AARM, the number refers to the text in draft 17 of the Ada 202x AARM. Paragraph numbers in other drafts may vary. Other paragraph numbers come from the final consolidated Ada 2012 AARM; again the paragraph numbers in the many drafts may vary.

Detailed Review of Ada 2012 AIs

AI12-0191-1/02 Clarify “part” for type invariants

Steve asks about the name of the concept. He is currently using “screened”. These are components that aren't known to the compiler, even when ignoring privacy. (That could even be components that don't exist yet in that they might be declared in future extensions.) This is not the same as visibility, so “hidden” is misleading.

Tucker doesn't like any of Steve's choices, specifically because this is not about visibility. He suggests “extended components” or “supplementary components”. Jeff suggests “extension components”. That doesn't work as it already has a meaning.

Steve thinks that “extended” is too close to existing terms. He thinks he will use “supplementary” for now, pending a better idea.

We've run out of time, so we quit without voting on anything about this AI.

AI12-0236-1/08 declare expressions

Tucker wonders if we should have a term for “new” limited objects (that is, the list of 7.5(2.1)). These are the objects that are required to be built-in-place. Tucker will take this off-line, and propose an AI.

Approve AI: 10-1-0.

Jeff still opposes, for the reason of putting the body in the specification, as discussed last time.

AI12-0262-1/09 Map-Reduce attributes

Gary asks if we write the name of the types out. Acc_Type in particular sounds like an access type of some sort. Tucker notes that Accumulator_Type is awfully long and hard to type correctly. Someone suggests that "Accum_Type" would be much clearer. We decide to add two more letters to each. Acc_Type => Accum_Type. Val_Type => Value_Type.

The AARM note that would now read "Value_Type is short for value type" needs to be changed. Leave that to the editor.

We go over other wording changes in detail.

Steve asks whether the Bounded Error is sufficiently defined. In particular, he wonders if "equal" is the user-defined equal or some predefined equal – this might matter if both exist.

Tucker thinks about this a bit and says that the user-defined equal seems OK. Should we add a AARM To Be Honest note to mention that "equal" means semantically equal; so a user-defined equal can be used? Yes.

Gary wonders why the combiner is allowed if **parallel** isn't present. Brad says that it makes it easier to switch back and forth between parallel and sequential operation. Tucker says that in a generic you might not know if the types are the same. Gary is satisfied with these answers.

Ed worries that "evaluating the value_sequence" implies that the values are produced then. Tucker notes that we're creating a stream and then passing it through to the operations.

Steve suggests a To Be Honest note to explain that "evaluated" is just organizing the sequence.

Ed suggests changes the word to "initialize" the value_sequence. That sounds like an object, and a sequence is not an object. "Priming" is then suggested. The term would have to be defined, but that could happen up above.

Someone should find a good word. Steve and Randy will try to come up with a word. "setting up" is iffy. Justin suggests "preparing". Tucker suggests "initiating".

Gary has a pile of typos.

- In !problem, "pre- and postconditions", rather than whatever is there.
- Also, "pre[ ]condition". "post[ ]condition". "more consise{,}"
- "...a shift in [a] the balance..."
- Double spaces after periods. (Gary didn't mention this one, I did – Editor.)
- In the paragraph starting: "Another need..." "divide{-}and{-}conquer".
- In !proposal "left{-}hand side ..."

Approve AI with changes: 10-0-1. Bob abstains.

AI12-0266-1/11 Parallel container iterators

Randy notes that the iterators with a Start parameter need some fix. The current wording completely ignores the Start parameter, which can't be what we want. We either need to reword that, drop the capability, or add a version with a Finish parameter. The latter seems like too much for right now. Tucker agrees that the iterators with a Start parameter are problematic for parallel execution, and doesn't think the capability adds much. Brad notes that splitting part of a container into chunks could be harder than splitting the entire container (where the number of nodes is known). We decide to drop this capability.

Tucker suggests that we drop the first "parallel" from the iterator introduction. The same change can be made to the "parallel and reversible" ones. In all of these cases, we describe how they work for each supported case, and the type also implies the operations.

In those same paragraphs, replace “and starting with all nodes concurrently when used as a parallel iterator” with “and processing all nodes concurrently when used as a parallel iterator”

Gary: Add after 5.5.2(10/3): “in an implementation{-}defined manner”. In the AARM Discussion “tree{-}based”.

Approve AI with changes: 8-0-2 Gary abstains, Bob abstains.

AI12-0306-1/01 Split null array aggregates from positional array aggregates

Steve wonders if the 2nd dimension can be a multidimensional null aggregate. Yes, there doesn't seem to be a reason to disallow:

```
        (1 .. 10 => [], 11..20 => (1 .. 0 => (1 .. 0 => 2)))
```

Jeff: The 4.3.3(42) changes is missing the open parenthesis at the start of in the first example.

Jeff also notes that the paragraph numbers change. It is OK for notes and later to change. The implementation permission is new (as is the second note), so the numbers after it shift.

Approve AI with changes: 9-1-0.

Bob is opposed: he still hates allowing array aggregates to have square brackets but not record aggregates. Randy points out that this is not introduced or changed by this AI. Someone notes that we asked for a reason when someone opposes an AI; it's not required for that reason to make sense. We move on.

AI12-0307-1/01 Resolution of aggregates

We discuss the choices, and agree that (3) is the way to go. The incompatibility isn't very likely, and if if does occur, it is always fixable with qualification (which, Randy points out, can be done directly on square bracket aggregates).

Brad: “(1) is unusual in {that} the syntax ...”

Approve AI with changes: 8-0-2. Gary and Jean-Pierre abstain.

AI12-0308-1/01 Declared entities need to be declarations

Steve: “declar{at}ive regions” in the !problem. Jeff: “list of declaration{s}”.

The subject “reference{d}[s]”. Or find a way to shorten it. [Editor's note: That was done in the name used above; the old one was too long for this page.]

Gary: “Add ", chunk_specification" in the comm{a}[ent]-separated list of 3.3.1(23/3),”

Approve AI with changes: 10-0-0.

AI12-0309-1/01 Missing checks for pragma Suppress

11.5(25.a) should say AARM 11(25.a), since it is an AARM note. (See the AI style guide recently posted.)

“Other language-defined checks that raise Program_Error:”

“a[n] class-wide” “{no} misuse ...” (two places). Spell “convertable” properly.

!discussion Brad: “This would work, but seems would be able to remember that”. This is a fragment of a thought. Probably replace by “this would be a lot of change”.

Next paragraph: “causal” should be “casual”.

In AARM 11.5(25.a) "{(those with names)}."

Steve suggests swapping the order (putting implementation-defined first). Bob would prefer to just drop that sentence, as discussing what implementation-defined stuff must do is always bogus. Let the editor decide something.

Approve AI with changes: 7-1-2.

Bob is opposed: It is not a serious problem. And he doesn't agree with the conclusion in the AARM note. Ed abstains. Justin abstains.

AI12-0310-1/01 Specifying the private parts of a packages in aspect Global

Tucker would like a !example showing this with the new wording. (or in the !discussion).

Steve wonders if any wording needs to be changed for this change. We think not, until Tucker notices that it says "follows" in paragraph 33/5. So that needs to say "if reserved words "private of" precede the package name".

Approve AI with changes: 10-0-0.

AI12-0311-1/01 Suppressing client-side assertions for language-defined units

Steve would like to reduce the duplication in the wording, preferably down to just a list of ancestor units. Randy doesn't think it would make much sense that way. Steve is given the AI to attempt to reduce the needed wording.

Approve intent: 9-0-1. Bob abstains.

AI12-0312-1/00 Examples for Ada 2020

We need a volunteer to look at the examples in the Standard, to see if there are appropriate examples for new features. Randy notes in particular that there are too many examples for container aggregates, and other things have none (he invented one for declare expression, for instance).

Tucker looks at the 4.3.5 examples and says that he intended those only for the AI and didn't mean to put them into the !wording.

Brad volunteers to look at the examples. Ed will lend a hand. Jeff offers to review. (Brad will lead.)

AI12-0313-1/00 Introduction for Ada 2020

Randy notes that the "Changes from the Third Edition" is an ISO-required part of the Introduction, so we need to have something written. It would be good to go over the entire Introduction, especially the Design Goals and Language Description sections to see if there is anything that should be covered.

Steve and Tucker volunteer (Steve is the lead).