Minutes of Electronic ARG Meeting 62U

30 March 2023

Attendees: Steve Baird, Randy Brukardt, Arnaud Charlet (left 13:05), Jeff Cousins, Gary Dismukes, Claire Dross (left 12:51), Bob Duff, Brad Moore, Jean-Pierre Rosen (left 12:45), Ed Schonberg, Justin Squirek, Tucker Taft, Richard Wai (joined 12:20).

Observers: Alejandro Mosteo (left 13:13).

Meeting Summary

The meeting convened on Thursday, 30 March 2023 at 10:36 hours EDT and adjourned at 13:29 hours EDT. The meeting was held using Zoom. The meeting covered about half of the agenda.

AI Summary

The following AIs were approved:

AI12-0454-1/01 Last second editorial fixes for Ada 2022 (12-0-0)
AI22-0048-1/02 Time_Zone parameter for Day_of_Week (12-0-0)
AI22-0068-1/01 Fix AI22-0028-1 fix to leave lead-in alone (12-0-0)

The following AIs were approved with editorial changes:

AI22-0017-1/01 Objects declared in declare_expressions may be surprisingly long-lived (12-0-0)
AI22-0052-1/03 Storage Pool-related side effects (12-0-0)
AI22-0060-1/01 Clarify interactions between defaults and check suppression (11-0-1)
AI22-0061-1/01 Assertion policy for duplicated expressions (12-0-0)
AI22-0070-1/01 Cannot use might, might use can (12-0-0)

The intention of the following AIs were approved but they require a rewrite:

AI22-0051-1/03 Preelaborable_Initialization and contract aspects (11-0-1)
AI22-0053-1/03 An unintended consequence of AI12-0101-1 (12-0-0)
AI22-0062-1/01 Clarify “ceases to exist” definitions (13-0-0)
AI22-0067-1/01 The nominal subtype of an aggregate (11-0-0)
AI22-0069-1/01 Empty subsequences in parallel reduction expressions (13-0-0)

The following AIs were discussed and assigned to an editor:

AI22-0033-1/01 Additional terms and definitions
AI22-0055-1/02 Usage Advice
AI22-0063-1/01 Font alone should not differentiate terms
AI22-0064-1/01 Obvious rules that should be stated
AI22-0065-1/00 Specialized Needs Annexes should be normative

Detailed Minutes

Welcome

Steve welcomes everyone to this meeting.

Apologies

Richard Wai will be late (he has another meeting). Tullio Vardanega has a work conflict and cannot attend.

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 12-0-0.

Date and Venue of the Next Meeting

Our next meeting is the hybrid meeting in Lisbon, Portugal in association with the Ada-Europe conference. It will be scheduled for June 11-13 with meeting starting at 9:00 local time each day and ending at noon on the 13th (the WG 9 meeting will be in the afternoon of the 13th).

Our next electronic meeting will be scheduled during that meeting.

FDIS Progress Report

Randy gives a short report. We got a "proof for checking", which was primarily the same as the last version we submitted. There were a handful of things that needed fixing, those were returned about March 10th to Bill Ash, who said he returned it to them. The status page hasn't changed since then; it gives a projected publication date of April 27th. We'll see.

There were a few editorial changes that needed to be applied to the RM. Those are outlined in AI12-0454-1 (see below).

UCI Working Group Report

Richard was late, so we deferred this report until he joined the meeting.

He said that after using the process for a while, we had identified some murky areas in the process. We convened a UCI Working Group meeting to discuss and resolve them.

The full process workflow could be too heavyweight for simple or small corrections. Some simple amendments could potentially be handled by an individual rather than a study group.

One of the work products of the meeting was to revise the ARG procedures so that the ARG can vote to send a proposal to prototyping or to a language study group as needed.

Richard is intending to write up additional details about the ARG process based on the updated ARG procedures to hopefully clarify the triggers for transitions between stages.

Updated ARG Procedures

One of the work products of the UCI meeting was some small updates to the ARG procedures. We added two new status possibilities and revised others. In particular, Approve Intent for Prototyping and Further Study Required. For the first, the AI is assigned to an ARG member to work with an Ada implementer to produce a prototype implementation including tests in order to get some experience with the feature. For the second, someone (ARG or a proposer) is assigned to create a language study group to further develop a proposal and report back to the ARG when that is complete.

We review the detailed changes, no needed corrections were identified.

Approve ARG Procedures 3.2: 12-0-0.

Unfinished Action Items

There are a few unfinished action items (Tucker Taft: OpenMP Technical Report; AI22-0034-1, Edward Fish: AI22-0022-1). Richard Wai had not finished AI22-0049-1 when the agenda was created, but he has done so since. We did not discuss these. Additionally, two people still owe ACATS Tests: Ed Schonberg and Justin Squirek.

Current Action Items

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

Steve Baird:

- AI22-0065-1 (with Tucker Taft and Randy Brukardt)

Randy Brukardt:

- AI22-0062-1
- AI22-0064-1
- AI22-0065-1 (with Steve Baird and Tucker Taft)

Editorial changes only:

- AI22-0017-1
- AI22-0052-1
- AI22-0060-1
- AI22-0061-1
- AI22-0070-1

Jeff Cousins:

- AI22-0033-1

Edward Fish:

- AI22-0022-1

Ed Schonberg:

- ACATS C-Test(s) for filters

Justin Squirek:

- ACATS C-Test(s) for Wide versions of Command_Line, etc. (or choose another, untested area)

Tucker Taft:

- OpenMP Technical Report.
- AI22-0034-1
- AI22-0051-1
- AI22-0053-1
- AI22-0055-1
- AI22-0063-1 (Split work with Richard Wai)
- AI22-0065-1 (with Randy Brukardt and Steve Baird)
- AI22-0067-1
- AI22-0069-1

Richard Wai:

- AI22-0063-1 (Split work with Tucker Taft)

Detailed Review

The minutes cover detailed review of Ada 2022 AIs (AI22s). The AI22s 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 2022 AARM, the number refers to the text in draft 35 of the Ada 2022 AARM (the submission draft). 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

AI12-0454-1/01 Last second editorial fixes for Ada 2022

Since Randy provided this late (about 9 hours ago), we reviewed each of the changes in detail. There were no comments on the changes.

Jeff noticed some funny characters in the AI. Randy thinks that those are the UTF-8 encoding of the En-dash (which is not an ASCII character). He probably cut-and-pasted those from the Word document without noticing, and modern Windows switches to UTF-8 when that happens. He'll replace those with hyphens in the file.

Approve AI: 12-0-0.

Detailed Review of Ada 2022 AIs

AI22-0017-1/01 Objects declared in declare_expressions may be surprisingly long-lived

Steve explains the AI. There is no semantic problem, it just might be surprising that finalization happens later than the declare_expression. So we add an AARM note to that effect.

Brad and Tucker are confused by the first paragraph of the discussion. We discuss the second sentence for a while but eventually decide it is correct (perhaps a bit hard to parse). The first sentence is incorrect, however. Randy and Steve suggest fixes but Tucker doesn't understand them. We decide to defer that to Editorial Review. Randy should suggest a rewrite for the paragraph and circulate to Steve, Tucker, and Brad.

Approve AI with changes: 12-0-0.

AI22-0033-1/01 Additional terms and definitions

Jeff says he would really like definitions of these. He is convinced to take the AI.

Gary notes the !ACATS says "{so no}[not] tests are needed".

Keep alive: 11-0-0.

AI22-0048-1/02 Time_Zone parameter for Day_of_Week

Tucker checked what GNAT does, and it uses local time. Randy notes that the same holds for Janus/Ada, he didn't get to check ObjectAda. Since local time is the most common interpretation, a defaulted parameter won't work (as that would provide the result in UTC time).

Approve AI: 12-0-0.

AI22-0051-1/03 Preelaborable_Initialization and contract aspects

Tucker explains his proposed wording.

"Suppressed" should be "ignored", since that is the term used. Suppressed would imply that failed checks would make the program erroneous. Jean-Pierre suggests using "omitted"; that seems to read better.

Brad notes that "D_I_C" should be introduced in the !issue. It's used before it is introduced.

Tucker will finish the AI using this wording.

Approve Intent: 11-0-1. (Gary abstains.)

AI22-0052-1/03 Storage Pool-related side effects

Steve explains the AI.

Brad notes a typo in the example. “My_Coll_Storage_Pool” should be “My_Cool_Storage_Pool”.

Approve AI with changes: 12-0-0.

AI22-0053-1/03 An unintended consequence of AI12-0101-1

Tucker explains the AI and the update.

Tucker worries that the profile of the redefined “=” could have a different subtype. That would be weird as clients could get a mysterious exception from a constraint that they cannot see. We need a Legality Rule to add a requirement for subtype conformance in this hidden case. Steve wonders if this is a problem for visible “=”, too. It would be similar, but it is visible to the client and it would be incompatible to reject it. Randy suggests that we not add any unnecessary incompatibilities, since those didn’t work in the past. (Recall that the entire reason for AI12-0101-1 was to repeal legality checks for unusual equality operators.)

Approve intent: 12-0-0.

AI22-0055-1/02 Usage Advice

This is an attempt to reorganize to meet ISO requirements that notes don’t say “should”.

Tucker says he only made the minimal change at this point. He wants to check the context and possibly move some of the examples under “Usage” (for instance, the examples using the random numbers to simulate dice rolls and coin tosses).

Randy notes that he had suggested to revert some of the wording to the earlier wording using “should”. Tucker shows an example of that in 9.8(22/5): the current wording is “is best used”, and it probably should just say “should”.

Randy also notes that we need a description of the Usage subheading in 1.1.2.

Tucker will keep the AI.

Keep alive: 10-0-0.

AI22-0060-1/01 Clarify interactions between defaults and check suppression

Steve starts out explaining pragma Assert, which is not this AI. Steve and Tuck explain *this* AI.

The AARM Note seems irrelevant to the RM. It should be moved to the !discussion.

Approve AI with changes: 11-0-1 (Jeff abstains).

AI22-0061-1/01 Assertion policy for duplicated expressions

Steve explains the AI (again, since he already explained it by mistake during AI22-0060-1).

Tucker questions “pairwise”. Steve notes that “full conformance” is a relation on two items, so if there are more than two, it needs to say something additional. It is felt to be overly pedantic, and the word is struck.

Brad worries that “first” is too vague. Tucker notes that the wording says “place of first expression”. We could say “place of first occurring expression”. Steve now complains that we’re complaining about *him* being too pedantic. We decide not to change this.

Approve AI with changes: 12-0-0.

AI22-0062-1/01 Clarify “ceases to exist” definitions

Steve describes the AI. Randy had complained this needs normative wording, and Steve and Tucker now agree. And they also suggest assigning the AI to Randy.

Approve intent: 13-0-0.

AI22-0063-1/01 Font alone should not differentiate terms

Randy explains having terms with different meaning only differentiated by font is confusing and violates principles of “accessibility” for the visually impaired. Steve mentions that it makes conversations and e-mails awkward, as it is hard to indicate the font when talking or writing in plain text.

Bob notes that requiring everything to be different from reserved words isn’t tenable. We need to be able to say “procedure”. Steve mentions “is”. Randy notes that they often (but not always) are preceded by “reserved word”, so they can be differentiated that way. We agree that we don’t need to make reserved words differ from other terms.

Steve says the he runs into “aspect specification” vs “aspect_specification”. The former includes pragmas (according to him). Other people agree and disagree with the premise, but this one should be checked.

Stage one of updating this AI is identifying the problematic pairs of terms (syntactic and semantic), and suggesting alternatives for at least one. Randy has shown examples for a couple of terms that he has verified are problematic this way.

The next step will be to decide (at a meeting) on a specific suggestion for each case, and only then develop the exact wording changes needed.

Tucker and Richard will take this AI and split up the work.

Keep alive: 10-0-0.

AI22-0064-1/01 Obvious rules that should be stated

Randy notes that we’ve defined all of the complex semantics for parenthesized expressions, so we should define the obvious stuff.

We look at the rules carefully, no comments occur.

We then turn to the current instance rule.

In the current instance rule, “or discriminant” is redundant. Steve notes that in a record representation clause, we don’t want the component name to be included. So we can’t say “declarative region”, Tucker thinks that just inside of a type_declaration is enough. [Editor's note: That would not include a protected body, the originating case of this question, so it clearly is not enough.]

Randy then wonders if a single_protected_declaration is covered with this wording. Tucker doesn’t think it needs any wording, it is “obvious” that the component is referring to the single object. Randy says that the whole point of making this wording explicit is that nothing is really that obvious as there is no prefix in this case. Tucker says that it is equivalent to obj_name.component, Randy again says that nothing in the RM says that. Steve jumps in and agrees with Randy, but that doesn’t seem to progress the discussion any. We agree to try this again later.

Split this into two AIs (parens and current instance). The parens one is essentially finished, the second is a keep alive. [Editor's note: The second AI, having the current instance rule, is AI22-0072-1.]

Keep alive: 12-0-0.

AI22-0065-1/00 Specialized Needs Annexes should be normative

ISO thinks that these annexes are “Informative” because they only contain “optional requirements”. Tucker objects to that as they are really normative (instructions to users and implementers that have the force of a standard if implemented). He would like to rewrite the introductions to make them normative to ISO.

ISO considers Annex B as normative because it is not completely optional; there are a few required elements at the start of the annex, even though the bulk of it is optional or implementation-defined. That provides a way forward for the SNAs.

Someone wonders if it really matters. We can keep the normative designation in the RM and let ISO do whatever they want. It’s best to keep the RM and ISO standard as close as possible; differences that we can eliminate without harming usability should be removed.

Randy notes that 18009, the ACATS standard, is OK with the annexes being informative. He had been concerned that the ACATS wouldn’t be able to test “informative” material. However, 18009 only talks about testing requirements, and even mentions the possibility of optional requirements. So the normative/informative marker has no bearing on that (thank goodness).

Gary notes a typo in !ACATS Test “do” should be “does” in the first sentence.

Randy, Tucker, and Steve will try to work on this and make some sort of concrete proposal.

Keep alive: 10-0-0.

AI22-0067-1/01 The nominal subtype of an aggregate

We discuss this. There is an obscure case involving aggregates of unchecked union types. The definition of “inferable discriminants” requires the object in question to have a constrained nominal subtype. Since aggregates don’t have a defined nominal subtype, it hardly can be constrained, so almost any use of an aggregate of an unchecked union type raises Program_Error. This is surprising since the discriminant is right there in the aggregate, it should be easy to infer the discriminant value. Jeff notes that this problem bit him writing ACATS tests; one wants to compare a calculated result against the expected result but doing so raises Program_Error.

Tucker notes that every aggregate is constrained. So it would make sense for the nominal subtype to be constrained.

Steve worries that the constraint could be dynamic, it might require extra code in the case of unchecked union where the discriminants aren’t stored. (Recall that we now allow discriminants to be dynamic so long as the subtype selects a single variant. We can fix that separately to require the discriminants to be static for unchecked union, or possibly some other fix.

Tucker will take the AI.

Approve intent: 11-0-0.

AI22-0068-1/01 Fix AI22-0028-1 fix to leave lead-in alone

Randy had noticed that the change in AI22-0028-1 had orphaned 4 other bullets. We need to make a much smaller change.

We look at the total effect of the two changes.

Approve AI: 12-0-0.

AI22-0069-1/01 Empty subsequences in parallel reduction expressions

Randy realized that there was a clever reading of the current wording which answers the question posed. This avoids requiring extra overhead and lets implementers use the most efficient possible code.

Unfortunately, Tucker notes that with filters, a non-empty chunk could generate an empty subsequence. So we do need normative wording to handle this case.

Tucker will take the AI.

Approve intent: 13-0-0.

AI22-0070-1/01 Cannot use might, might use can

Randy explains that AI22-0006-1 uses “might” in its wording. That happened before we realized that “might” is not allowed. Since AI22-0006-1 is WG 9 Approved, we need a separate AI to fix this up. We had a similar problem in AI22-0031-1, but that AI is in editorial review so it should just be corrected without ado.

Brad thinks that the “can” earlier in this wording paragraph should be “may”. Randy checks and determines that this is indeed an Implementation Permission, and therefore it certainly should be using “may” and indeed it is weird that is does not. We change the first “can” in the paragraph to “may”.

Approve AI with changes: 12-0-0.