Minutes of Electronic ARG Meeting 62R

8 September 2022

Attendees: Steve Baird, Randy Brukardt, Jeff Cousins, Gary Dismukes, Claire Dross (joins at 12:26), Brad Moore, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega.

Observers: None.

Meeting Summary

The meeting convened on Thursday, 8 September 2022 at 10:34 hours EDT and adjourned at 13:33 hours EDT. The meeting was held using Zoom. The meeting covered much of the agenda.

AI Summary

The following AI was approved:

AI22-0040-1/02 Finalization and implicit loops (10-0-0)

The following AIs were approved with editorial changes:

AI12-0446-1/01 Additional wording improvements (9-0-0)
AI22-0024-1/02 Abstract prefixed views (8-0-1)
AI22-0043-1/01 Look through more than parens... (8-0-2)
AI22-0045-1/02 Issues with pragma placements (10-0-0)
AI22-0047-1/01 Initialization of and saving ... (9-0-0)

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

AI22-0036-1/03 Attributes in the expression of Default_Value (9-0-1)

The following AI was discussed and assigned to an editor:

AI22-0009-1/03 Nonvolatile views of volatile objects

Detailed Minutes

Welcome

Steve welcomes everyone.

Apologies

It's John Barnes' diamond wedding anniversary, so he is otherwise occupied. Claire Dross, Arnaud Charlet, and Richard Wai all have conflicting meetings (all different meetings). Jean-Pierre Rosen is on vacation.

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 9-0-0.

Date and Venue of the Next Meeting

Randy proposes Thursday, November 10, 2022 for our next meeting.

There are no objections to the November 10th date; so we adopt that date. The meeting time and duration remains the same.

FDIS Progress Report

ISO bounced the FDIS, wanting a lot of reformatting this (and a few wording changes, found in AI12-0446-1 that we will discuss shortly). Randy is changing the formatting tool to do this reformatting while still allowing the RM format to be mostly unchanged.

This a general problem; other language standards are encountering similar problems (and don't have a formatting tool to change). Complaints about this during the most recent US SC 22 tag meeting resulted in the US making a formal request [Editor's note: primarily drafted by Pat Rogers based in part on my suggestions] to the full SC 22 for a resolution requesting change at the JTC1 level. While this feels good, it is unlikely to have any constructive results in time for this go-round.

Randy expects to have this completed next week; he'll also post a new RM draft at that time (mostly with some typos fixed).

Working Group Report – UCI (User Community Input)

In Richard's absence, Tucker gives a report. He says that we're mostly done and moved over to the new processes. There has been quite a bit of activity on the Github forum in the past few weeks, so it appears to be taking hold.

Jeff notes that there hasn't been an announcement. He thinks there should be one. It should be in (at least) the Ada User Journal, AdaIC, Ada-Comment, and comp.lang.ada. Tucker is given an action item to write such an announcement.

[Editor's note: I should have noted that I haven't made much progress in completing the AI conversion, as I have been working (again) on the FDIS. It is still planned to convert all of the existing AI22s to the new HTML format.]

Processing Amendments and Hold AIs

We've had some requests to restart some of the Ada 2012 Hold AIs. There are 41 such AIs, of varying importance, difficulty, and completeness of proposal. We need to decide how to decide what to promote and what to leave on indefinite hold.

Randy suggests we start by deciding what sort of Amendments that the ARG should process immediately. He thinks we shouldn't do big things, without WG 9 instructions and without Ada 2022 being finalized or fully implemented by anyone. (Of course, we can create language study groups for significant new features, but not put them on the ARG agenda.) On the other hand, there are various simple issues that should be handled now. For instance, we have a Github issue that says that Day_of_Week needs a time zone parameter. It's hard to imagine what value there is to waiting to process such things (and processing them will reduce the backlog when we do start working on large amendments).

Tucker suggests that we immediately handle AIs that are fixing holes and don't require prototyping (since we've committed to trying to prototype most significant new features). We'll take a relatively expansive meaning of "holes" (the missing class-wide user-defined literals qualify, for instance). That notion seems to get general agreement.

Turning back to hold AIs — someone needs to categorize the AIs into categories (fixes to do now, ideas we probably want to pursue, and ideas that we can let drop). It is suggested that in the past for similar kinds of things, we have had the chair and the editor do a proposal, and then let the full group discuss (possibly making changes) and then finally vote on it.

So we ask: Randy and Steve to categorize the hold AIs, and then we will have a more general discussion on the proposed categorizations, and then a letter ballot on which ones to promote now and which ones become Github issues for future language study groups.

Unfinished Action Items

There are a few unfinished action items (Tucker Taft: OpenMP Technical Report; AI22-0034-1; Edward Fish: AI22-0022-1). We did not discuss these. Additionally, three people still owe ACATS Tests: Ed Schonberg, Justin Squirek, and Tucker Taft.

Current Action Items

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

Steve Baird:

- AI22-0009-1
- AI22-0036-1
- Work with Randy Brukardt on a proposed categorization for AIs that are currently on hold.

Randy Brukardt:

- Work with Steve Baird on a proposed categorization for AIs that are currently on hold.
- Assist Tucker Taft in wordsmithing 4.1.3(9.2/3) (see discussion of AI22-0024-1), with the help and review of Gary Dismukes.

Editorial changes only:

- AI12-0446-1
- AI22-0024-1
- AI22-0043-1
- AI22-0045-1
- AI22-0047-1

Gary Dismukes:

- Assist Tucker Taft in wordsmithing 4.1.3(9.2/3) (see discussion of AI22-0024-1), with the help and review of Randy Brukardt.

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:

- AI12-0346-1 – also, construct the Technical Report suggested by this AI.
- AI22-0034-1
- Wordsmith 4.1.3(9.2/3) (see discussion of AI22-0024-1), with the help and review of Randy Brukardt and Gary Dismukes.
- Create an announcement of the new UCI process and web sites (see UCI WG report).

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 33 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

AI12-0446-1/01 Additional wording improvements

Randy explains the problems that these changes address. Some of these are from the latest batch of FDIS comments, some are from recent editorial comments, and the rest are updating the "Instructions for comment submission". All will be applied to the Ada 2022 draft.

Tucker explains that the ISO editors take a very local view of wording. The RM does not use "shall" for rules that apply to implementations (except for "Implementation Requirements"), and we don't want to change that. Tucker believes that our current rules make the RM much more readable.

Randy notes that ISO comments rarely go past chapter 1, other than searchable things, which we've hopefully handled in our previous iteration. So once we get those things right...

Gary says that we need a "the" in 1.1.2(2):

{All implementations shall conform to the}[The]

Brad suggests in 0.2(19) that "becomes" should be "evaluates to".

Ed notes "can may" in this paragraph. Delete the "may" (that is already the case, it was copied in plain text without the deletion markings).

Approve AI with changes: 9-0-0.

Detailed Review of Ada 2022 AIs

AI22-0009-1/03 Nonvolatile views of volatile objects

Steve explains the wording.

Tucker asks about the recommendation, which causes us to read the summary, which does not reflect the actual wording (which makes a copy). Tucker corrects the summary.

Steve and Tuck think the wording was changed from this, Randy does not recall any such proposal. We decide to defer this one to find the correct wording.

[Editor's note: As we didn't specify anything else, this goes back to Steve Baird, the last author of record.]

AI22-0024-1/02 Abstract prefixed views

Tucker tries to explain the problem. A prefixed view is a subprogram, and we need to define the properties of that subprogram in cases where the (expanded) call is dispatching and/or abstract. This matters when the prefixed view is used in a renaming or passed as a formal parameter.

Gary notes that "controlling (access) parameter" is confusing, he suggests mentioning that separately. Tucker writes a version that is more pedantic but is generally thought to be better.

Steve notes that there are two choices for the type of the parameter: it is currently the type of the prefix; but should it be the class-wide version of the parameter's type instead? Tucker thinks that would be weird; the type of prefix determines what declaration you are using, so the profile you would see is the one where all of the (controlling) types match the prefix. (That would sometimes be an inherited operation.) So Steve's suggestion would not actually

change the type of the parameter, it is clearer as Tucker has proposed it (since readers don't have to figure out how operation inheritance affects the parameter types).

The constraints come from the unprefixed profile. (These have to be first subtypes, but those can have constraints in derivations.)

Steve suggests that Tucker, Randy, and Gary wordsmith the new wording, but otherwise approve the AI.

These renamings can't be primitive as the prefix is an object, which would freeze the tagged type, and then the primitive would be too late for the type. For an untagged type, that could happen, but we don't have untagged prefixed views nor untagged class-wide types, so that's a problem that we don't need to solve.

Approve AI with changes: 8-0-1. Jeff abstains.

AI22-0036-1/03 Attributes in the expression of Default_Value

Attribute_definition should be attribute_reference in the !wording.

Does this allow 'First? Randy argues that it does because the scalar 'First is a different attribute than the array object 'First. Tucker thinks that argument is too subtle for most readers. Randy suggests a To Be Honest note to clarify. After some discussion, we decide to add "scalar" in two places in the wording. Tucker makes this change.

Brad would like a list of likely attributes. We add an AARM note to that effect.

Approve AI with changes: 8-0-1. Gary abstains.

We immediately have a short break. The original list of attributes mentioned in the new AARM note included Enum_Val and Enum_Pos. But those are problematic, since the type needs to be frozen to determine its representation. In contrast, First, Last, Val, and Pos don't depend on anything that is learned during freezing. But we realize that the current rules freeze the type of the prefix of all attributes, so that we've managed to allow nothing at all with this AI. (Such circular freezing is illegal.)

So this AI has to go back to Steve in order to consider interactions with freezing. Randy notes that the !discussion seems confused as the prefix of S freezes T, and that's what makes the example illegal. If he had written this example with a Default_Value aspect, that is currently illegal as well.

```
type T is (A, B, C) with Default_Value => S'Val(1);
     -- Illegal freezing of S (otherwise OK).
subtype S is T;
```

So the claim that there is not anything new in freezing here is incorrect (the special case currently is only for enumeration literals). We likely need some new freezing exceptions for some attributes.

Approve Intent: 9-0-1. Gary (still) abstains. (Claire has just joined, thus the different numbers from the previous vote).

AI22-0040-1/02 Finalization and implicit loops

Steve explains the problem. Temporary objects can "build-up" in an implicit loop.

Tucker then explains the majority of changes (essentially, inserting a new master). This is possibly incompatible (if someone depended on the "late" finalization in such cases), but that is unlikely (depending on the order of finalization is rare to begin with, and most implementations probably are doing something other than what the RM specifies anyway).

He then goes on to explain the other change; it allows cleaning up non-build-in-place temporaries.

Randy is concerned that this permission allows too much. Tucker says that the rule moving the owner of coextensions prevents problems. Randy is still dubious, but doesn't have a specific problem right now. [Editor's

note: The rule Tucker refers to only applies when a function temporary is “used in its entirety”, while the permission applies to any function temporary. This suggests that the permission is overly broad; the special rules almost never apply to the result of a container Reference function, while it appears that the permission, as written, would.]

Approve AI: 10-0-0.

AI22-0043-1/01 Look through more than parens...

Shorten the subject to: Tags from context for tag-indeterminate calls.

Tucker explains the problem and the solution (generalizing the wording).

Randy worries that this requires somehow figuring out what expression is the result of a conditional expression, whereas currently one always knows what expression one needs to get the tag from. Tucker and Steve think that this is not appreciably different than the current rules. Randy admits that his memory of how this part of his compiler code works is hazy (he hasn’t needed to change it in many years), and drops the subject.

Approve AI with changes: 8-0-2. Jeff, Claire abstain.

AI22-0045-1/02 Issues with pragma placements

Tucker explains the problems and his proposed solution. The primary thing that we need to know is what construct certain pragmas are “in the place of” so we can properly determine the properties of the pragma. We also are confused on whether pragmas are allowed in declare_expressions.

We discuss specifically the case of the Suppress pragma. Looking at the wording in 11.5, it is not a configuration pragma per-se, but can act as one in some cases. Claire notes that the “old notion” of configuration pragmas is more narrow than what he has here. Tucker says he was trying to add the notion of “pragmas usable as configuration pragmas” to handle the extensions for pragmas like Suppress. Claire notes that is itself a kind of pragma, so then the lead-in saying that there are only five types is a lie.

Tucker puts “pragmas usable as configuration pragmas” in italics, and adds it to the lead-in in parentheses. This satisfies Claire.

Jeff does not like the wishy-washy resolution of pragmas in declare_expressions. He wants to see fewer, not more, implementation-defined things as this affects portability. Randy notes that the reason that we didn’t want to require control pragmas in declare_expressions as they could be an implementation earthquake (implementations can currently process expressions whole without the “internal state” [restrictions, suppressions, etc.] of the compiler changing during that process; that would change if such pragmas could appear in declare_expressions.)

He would prefer we make a decision. Therefore, we decide that a non-executable pragma is not allowed in a declare_expression.

Ed would like this statement given in 4.5.9 for declare expressions, so Tucker puts it there as part of 4.5.9(7/5), and the previous proposed permission is deleted.

Approved AI with changes: 10-0-0.

AI22-0047-1/01 Initialization of and saving ...

The subject is too long: make it “Accumulator updates during reduction”.

The !subject is in the wrong font (Tucker fixes online).

The !issue is missing the original question about the accumulator being constrained by its initial value. Tucker adds one from the e-mail. He then goes on to add a mention of this to the recommendation, and to the AARM notes.

Jeff notes that the wording for the note following 4.5.10(24/5) has the last comma in a weird place. The !summary has the same wording, but with the comma in the correct place. Tucker fixes that in the AI.

Approve AI with changes: 9-0-0.