Minutes of Electronic ARG Meeting 62N

3 February 2022

Attendees: Steve Baird, John Barnes, Randy Brukardt, Arnaud Charlet (left at 13:00), Jeff Cousins, Gary Dismukes, Claire Dross, Bob Duff, Edward Fish, Brad Moore (left at 13:00), Jean-Pierre Rosen, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega, Richard Wai.

Observers: None.

Meeting Summary

The meeting convened on Thursday, 3 February 2022 at 10:32 hours EST and adjourned at 13:30 hours EST. The meeting was held using Zoom. The meeting covered much of the agenda.

AI Summary

The following AIs were approved:

AI22-0008-1/02 Nominal subtype of a delta aggregates (16-0-0)
AI22-0013-2/01 Pragma after a final label (16-0-0)
AI22-0016-1/01 Newly constructed objects in a declare expression (14-0-0)
AI22-0020-1/02 Order of elements in Aggregate aspect (15-0-1)

The following AIs were approved with editorial changes:

AI22-0004-1/03 Permissions of 4.1.4 and No_Implementation_Attributes (16-0-0)
AI22-0014-1/01 Predicates on inherited functions (15-0-1)
AI22-0021-1/02 Aggregate aspect resolution should not be too smart (16-0-0)
AI22-0023-1/01 Deferred presentation issues from the WG 9 review (16-0-0)
AI22-0031-1/02 Issues with dynamic evaluation of container aggregates (16-0-0)
AI22-0032-1/02 Abstract and class-wide container aggregates (16-0-0)

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

AI22-0022-1/01 Difficult example issues from WG 9 review (16-0-0)
AI22-0035-1/01 Add “not null” to some Ada.Strings subprograms (16-0-0)
AI22-0036-1/01 Attributes in the expression of Default_Value (14-0-2)

The following AIs were discussed and assigned to an editor:

AI22-0028-1/01 Program_Error for unchecked unit equality

The following AIs were discussed and voted No Action:

AI22-0013-1/01 Pragma after a final label (16-0-0)

Detailed Minutes

Welcome

Steve welcomes everyone.

Apologies

None.

Previous Meeting Minutes

There were no comments on the minutes: Approve minutes: 16-0-0.

Date and Venue of the Next Meeting

Randy proposes Thursday, May 5, 2022 for our next meeting. There being no objections, we adopt that date.

We still have an in-person meeting scheduled, subject to ISO regulations, on June 17-19, 2022, Ghent Belgium (in conjunction with Ada-Europe).

Post-Ada 2022 draft RM naming

Randy would like to have a suggested name for our next version of Ada, so that he can use that in constructing the future drafts of the RM.

Tucker suggests Ada 2y. We could also use Ada 3x, but that assumes that we don't do another version for ten years. The trend in programming languages is for more frequent standards (and we only waited 5 years between Ada 2005 (completed in 2007) and Ada 2012. John suggests Ada xy since we really don't know either number.

We take a straw poll on the first two choices. Ada 3x 3, Ada 2y 11, Abstain 1 (Bob). Randy is directed to use "Ada 2y" for the next version.

AI Format Regularization

One of the UCI WG recommendations is to use the same headers for all kinds of AIs, to simplify their construction and make it easier to change classifications. Currently, some of the headers depend on the class of the AI. For instance, confirmations use !response, binding interpretations use !recommendation, and amendments use !proposal.

After several rounds of discussion, we decided to recommend !issue and !recommendation for all AIs.

Use !issue and !recommendation for all AI classifications: Approve: 16-0-0.

ACATS Tests for Ada 2022

We've received a few additional tests. We still need them, so if you took an assignment, please get them finished.

Tucker asks that ACATS tests be on his ARG list of things to do. So we would like to have ACATS test assignments in the Action Item list. Steve asks if a general reminder there is enough. Tucker would prefer if each member have a separate list.

Working Group Report – UCI (User Community Input)

Richard gives an update on the group's work. The new web site and Github are coming along, in varying amounts of completion. We're far enough along that it would help for ARG members to comment. We're still working on some issues, including the ultimate AI format for long-term storage (we'll use Google Docs for active development). Randy notes that we never announced the website to the ARG list (just the UCI list), so most haven't seen it yet.

Tucker shared his screen and gives a tour of the website. (Find it at: https://sites.google.com/view/ada-rapporteur-group) We will make it a subdomain of AdaIC in the future (we won't use the current long-winded Google link permanently).

Arnaud has trouble accessing the site. Tucker suggests that he try logging out of his AdaCore Google account before trying; there is some problem associated with AdaCore accounts. Jean-Pierre reports that he has no problem accessing it.

Unfinished Action Items

There are two unfinished action items (Tucker Taft: OpenMP Technical Report; AI22-0034-1). We did not discuss these.

Current Action Items

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

Steve Baird:

- ACATS C-Test(s) for delta aggregates

Randy Brukardt:

Editorial changes only:

- AI22-0004-1
- AI22-0014-1
- AI22-0021-1
- AI22-0023-1
- AI22-0031-1
- AI22-0032-1

Gary Dismukes:

- ACATS C-Test(s) for user-defined literals

Claire Dross:

- ACATS C-Test(s) for declare expressions

Edward Fish:

- AI22-0022-1

Brad Moore:

- AI22-0036-1
- ACATS C-Test(s) for procedural iterators

Ed Schonberg:

- ACATS C-Test(s) for filters

Justin Squirek:

- ACATS C-Test(s) for Wide versions of Command_Line, etc.

Tucker Taft:

- AI12-0346-1 – also, construct the Technical Report suggested by this AI.
- AI22-0028-1
- AI22-0034-1
- AI22-0035-1
- AI22-0038-1
- AI22-0039-1
- ACATS C-Test(s) for container aggregates

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 202x AARM, the number refers to the text in draft 32 of the Ada 202x 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 2022 AIs

AI22-0004-1/03 Permissions of 4.1.4 and No_Implementation_Attributes

Tucker would like quotes around "-- N/A => Error" and "-- ERROR: " in the !ACATS Test section.

Arnaud is confused about the reason for these rules. We explain that this is about improving portability. There are permissions for implementations to add additional kinds of prefixes to certain language-defined attributes, with implementation-defined semantics. Other implementations may not have those attributes at all, or might have different semantics. So when the restriction is in effect, we want those extensions rejected as well as uses of language-defined attributes.

Approve AI with changes: 16-0-0.

AI22-0008-1/02 Nominal subtype of a delta aggregates

This AI was rewritten to make it clear that there is no such thing as a nominal subtype of an aggregate. There is no change to the proposed wording.

Approve AI: 16-0-0.

AI22-0013-1/01 Pragma after a final label

We'll use the other alternative.

No Action: 16-0-0.

AI22-0013-2/01 Pragma after a final label

Approve AI: 16-0-0.

AI22-0014-1/01 Predicates on inherited functions

Remove "with SPARK_Mode" from the example in the !question.

--@PREDICATE_CHECK:FAIL should be – Should fail a predicate check.

Approve AI with changes: 15-0-1. Abstain: Jeff.

AI22-0016-1/01 Newly constructed objects in a declare expression

A slice is not newly constructed, we need to cover that here.

Approve AI: 14-0-0.

AI22-0020-1/02 Order of elements in Aggregate aspect

This is very similar to the rule for pragmas (such as pragma Import).

Approve AI: 15-0-1. Jean-Pierre abstains.

AI22-0021-1/02 Aggregate aspect resolution should not be too smart

Ed Fish asks why nonlimited aggregates are not allowed. The mechanism is designed to add elements one at a time, a different interface would be needed for limited elements.

In 4.3.5(11): the dash is a non-ASCII character that looks weird on Tucker's computer.

In the first paragraph of !discussion, change "but" to "and". Better: "but for consistency, it was done throughout the language".

Approve AI with changes: 16-0-0.

AI22-0022-1/01 Difficult example issues from WG 9 review

Edward Fish takes this AI. Tucker notes that the !appendix has suggested solutions for some of the issues.

Approve intent: 16-0-0.

AI22-0023-1/01 Deferred presentation issues from the WG 9 review

(1) Has no comments.

(2) Jeff was the requestor; he thinks it is an improvement. No one disagrees.

(3) Closing star on Universal is a left paren in the wording. Tucker suggests dropping the "one of". Steve says that it is needed to make the list clearly an exclusive list. We decide to keep it.

Treat as a typo, move to AI12-0437-1.

(4) This clarifies why we're referring to 9.5.2. Looks good.

(5) Tucker would like to pull this into a separate AI for wordsmithing. "a minimally useful" → "an" in the wording; it was left-over from a brainstorming session to clear writers block. Assign it to Tucker. [This will be AI22-0038-1 – Editor.]

(6) It is unclear what the original author was trying to accomplish with this rule (These rules also apply...). Make this into a separate AI – assign to Tucker. [This will be AI22-0039-1 – Editor.]

Gary complains about the hyphen in "non-overridable" in the wording; he notes it doesn't appear in the RM in this wording.

(7) No interest in changing this. Remove it from this AI. Jean-Pierre notes that the question has an extra "should" in the last sentence. (The answer will be "No.").

(8) We should not be using "identifier" here, so this is an improvement.

(9) This a suggestion from John. Treat as a typo, move to AI12-0437-1.

(10) Another suggestion from John. Treat as a typo, move to AI12-0437-1. Jean-Pierre notes that the question has "onject".

(11) This needs to be updated. We should do this in Ada 2022. Move to AI12-0437-1, as it is an ISO rule mistake that should be fixed ASAP.

(12) Fix this in Ada 2022. Treat as a typo, move to AI12-0437-1.

Tucker has typos in !discussion. (6) "broaded" should be "broadened". (11) "superceded" should be "superseded". This leads to a discussion of this word; Gary says that the former spelling is OK, the latter spelling is preferred.

Jeff: (8) “talk {of} a usage” or possibly “mention”.

Summary: (3), (9), (10), (11), (12) are treated as typos, thus will be moved to AI12-0437-1. (5), (6) to separate AIs. (7) no change at all (question stays here). (1), (2), (4), (8) stay with this AI, as does (7), with no change made for it.

Approve AI with changes: 16-0-0.

AI22-0028-1/01 Program_Error for unchecked unit equality

If a user-defined equality is used, the use of an unchecked union subcomponent isn’t a problem. The rules didn’t take into account composition of equality.

The goal of this rule is that it would be determinable at compile-time whether or not Program_Error would be raised (outside of shared generic bodies). That was considered important for safety.

Claire asks if the rule covers the case where the parent type uses predefined equality, and an extension uses user-defined equality. She would prefer the simpler rule that touching predefined equality raises Program_Error (this is too much of a corner case).

Tucker says that the idea is that you don’t need to look at discriminants or variants to determine whether there is a problem.

Ed Fish notes that extra safeguards when interfacing with C are a good thing.

Straw poll: 5 for delete 24/2 (fully dynamic rule). Improved version of AI: 2. Abstain 7.

So we will delete 24/2: Tucker will rewrite along these lines.

[Editor's note: A reviewer of the minutes noted that the subject of this AI has a typo - “unit” should be “union”. The minutes were not changed as they accurately reflect the (incorrect) subject of the AI; hopefully, this note will serve ensure that the subject gets corrected.]

AI22-0031-1/02 Issues with dynamic evaluation of container aggregates

In question (1), “contraints”.

In wording (2), we’re missing the closing } (should come after “of the aggregate”).

In wording (3), container_element_assoc{i}ation as the third parameter;

In discussion (1), paragraph 2, “Paramete”.

In discussion (2), “...to the order in which the the calls are made, just to the order of the...”

Approve AI with changes: 16-0-0.

AI22-0032-1/02 Abstract and classwide container aggregates

Wording changes:

... {subprograms}[names] specified in the Aggregate aspect for T shall be abstract.

Ed Fish notes that class-wide aggregates could be useful. Randy agrees (they could be used to assign to an in out class-wide parameter, for instance). Tucker notes that using [] to set something to empty seems useful as well. But Randy also notes that class-wide aggregates would clearly be an extension; we would need a new mechanism to determine the specific type tag of the aggregate (probably in the same way that tag-indeterminate functions do that). So, it should be submitted to the user community input process.

Gary: Need hyphen in class{-}wide in the subject.

Gary: Extra d in nonoverridable (twice).

Tuck: Anonymous is misspelled in the second paragraph of the discussion.

Steve asks about the need for the generic boilerplate. Randy notes that a non-abstract type can be the actual for an abstract derived type. Steve wonders about a derivation in a generic body. A usage in a generic body is not a problem, since it is abstract and no aggregate can be written, even if the actual is not abstract. But the exported version from a specification could be used by a client (if the actual is not abstract).

Brad: In !discussion (2), “for nonabstract tagged” (needs “type”).

The “One could argue that” paragraph has too many subjunctives. “One could argue that [if] following…” The next sentence “imply an{y} Legality Rules or Static Semantic rules”.

Remove this paragraph rather than trying to fix it; most find it more confusing than helpful. (The author having to explain what it meant; most readers will not have the author handy to ask.) The next paragraph should start “We also have...” rather than “Moreover”.

Next paragraph: “nonoveriddable”; it’s spelled correctly in the previous usage one line above.

Approve AI with changes: 16-0-0.

AI22-0035-1/01 Add “not null” to some Ada.Strings subprograms

We discuss whether this should be an Amendment or a Binding Interpretation. There is a minor incompatibility with this change: places where subtype conformance is required would become illegal. Note that normal renames and generic formal parameters would not be affected; only ‘Access and renames-as-body would be.

It is suggested that it is annoying to have language-defined libraries with subtly different specifications in different language version (and hard to implement). The speaker thought it would be better to just take the incompatibility across the board.

We take a straw poll on this question: Binding Interpretation: 11 Amendment: 4 Abstain: 1

Assign AI to Tucker.

Approve intent: 16-0-0.

AI22-0036-1/01 Attributes in the expression of Default_Value

We discuss the problem a bit. We decide that this is an Amendment. Bob would prefer not to have another if in GNAT. There is no incompatibility associated with this idea – it is a pure extension. So we quickly change our mind.

Approve intent: 14-0-2. Jeff, Ed S. abstain.

Brad will take the AI, Tucker will help as needed.