# Minutes of Electronic ARG Meeting 62T

19 January 2023

**Attendees**: John Barnes, Steve Baird, Randy Brukardt, Arnaud Charlet (left 13:32), Gary Dismukes, Claire Dross (left 13:05), Bob Duff, Edward Fish, Brad Moore, Jean-Pierre Rosen, Ed Schonberg, Justin Squirek, Tucker Taft, Tullio Vardanega (joined 11:07), Richard Wai (joined 12:10).

**Observers**: None.

## Meeting Summary

The meeting convened on Thursday, 19 January 2023 at 10:34 hours EST and adjourned at 13:36 hours EST. The meeting was held using Zoom. The meeting covered some of the agenda.

### AI Summary

The following AIs were approved:

AI12-0447-1/02 More rewordings of notes (13-0-0)
AI12-0448-1/02 Fixes for Terms and Definitions (13-0-0)
AI12-0449-1/03 Appearance of cross-references (13-0-0)
AI12-0450-1/01 Update references to standards (13-0-0)
AI12-0451-1/03 Still more changes to wording (13-0-0)
AI12-0452-1/01 Note on examples is not a thing (13-0-0)
AI12-0453-1/02 Remove “i.e.” and “e.g.” (13-0-0)

The following AIs were approved with editorial changes:

AI22-0036-1/06 Attributes in the expression of Default_Value (8-0-5)
AI22-0054-1/01 Ambiguous prefix for the Result attributes (15-0-0)
AI22-0058-1/01 Preconditions for checking Task_Ids (15-0-0)

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

AI22-0049-1/01 Seconds function with Time_Zone parameters (12-0-1)
AI22-0051-1/02 Preelaboratable_Initialization and contract aspects (14-0-0)

The following AIs were discussed and assigned to an editor:

AI22-0048-1/01 Time_Zone parameter for Day_of_Week
AI22-0052-1/01 Storage Pool-related side effects
AI22-0053-1/02 An unintended consequence of AI12-0101-1
AI22-0055-1/01 Usage Advice

## Detailed Minutes

### *Welcome*

Steve welcomes everyone to the first ARG meeting of 2023.

### *Apologies*

Richard Wai and Tullio Vardanega will be late (they both have other meetings). Jeff Cousins cannot attend.

### *Previous Meeting Minutes*

There were no comments on the minutes: Approve minutes: 13-0-0.

### *Date and Venue of the Next Meeting*

Randy proposes Thursday, March 30, 2023 for our next meeting. This is our usual 10 weeks between meetings.

There are no objections to the March 30th date; so we adopt that date. The meeting time and duration remains the same.

There will be hybrid meeting in Lisbon, Portugal in association with the Ada-Europe conference. Tucker has a conflict after the conference, so we suggest meeting before the conference. Tucker was thinking June 11-13 to overlap the meetings with the initial tutorial day of the meetings. Randy had suggested June 10-12, as he remembers conflicts and pushback from the conference organizers when we overlapped the meetings with the tutorial day in the past. In either case, we'll reserve the afternoon of the last day for the WG 9 meeting, which also will be hybrid. The conference schedule has not yet been set, so it is premature to worry about conflicts. We decide to give Steve an action item to set the dates in conjunction with Pat Rogers and an appropriate person at Ada-Europe (starting with Tullio Vardanega).

### *FDIS Progress Report*

Randy gives a short report. The FDIS was resubmitted last week. At this time, there has been some pushback on the format of the submission. The format was explained again by Bill Ash and we're waiting for a response. We have a group of AIs to vote on in a moment to approve the changes made. We're hopeful that this will be the last time; we're approaching the limit date for this project, so there is pressure on both sides to get this done.

### *Unfinished Action Items*

There are a few unfinished action items (Tucker Taft: OpenMP Technical Report; AI22-0034-1, announcement of the new UCI process and web sites; Edward Fish: AI22-0022-1). We did not discuss these. Additionally, two people still owe ACATS Tests: Ed Schonberg and Justin Squirek. (Jeff finished the partially completed tests Tucker had constructed, so he's off of this list.)

### *Current Action Items*

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

Steve Baird:

- AI22-0052-1
- Determine the dates for the hybrid ARG meeting associated with the Ada-Europe conference, coordinating with Pat Rogers and the Ada-Europe organizers.

Randy Brukardt:

Editorial changes only:

- AI22-0036-1
- AI22-0054-1
- AI22-0058-1

Edward Fish:

- AI22-0022-1
- AI22-0051-1 (with assistance from Tucker)

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
- AI22-0048-1
- AI22-0051-1 (assist Edward Fish)
- AI22-0053-1
- AI22-0055-1

Richard Wai:

- AI22-0049-1

## *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*

**AI12-0447-1/02 More rewordings of notes**
**AI12-0448-1/02 Fixes for Terms and Definitions**
**AI12-0449-1/03 Appearance of cross-references**
**AI12-0450-1/01 Update references to standards**
**AI12-0451-1/03 Still more changes to wording**
**AI12-0452-1/01 Note on examples is not a thing**
**AI12-0453-1/02 Remove "i.e." and "e.g."**

We had a review period via e-mail last month on these AIs. Some comments were received and incorporated in these AIs. The material has been already added to the FDIS, so any substantive changes will have to wait until Ada 202y.

Approve AIs: 13-0-0.

### *Detailed Review of Ada 2022 AIs*

#### AI22-0036-1/06 Attributes in the expression of Default_Value

Tucker modified the freezing rules to add some exceptions for this case. He also rearranged them. Tucker explains these in detail. There is a split between rules for constructs that "cause freezing" and rules for which entities are "frozen" when freezing is caused. He says Bob originally designed this split for Ada 95. Bob says that he couldn't get a handle on this problem and he gave it back to Tucker who designed these rules. This causes Tucker some concern; Randy notes that occurred over 30 years ago so memories aren't perfect.

Anyway, Tucker rearranged the rules so that all of the rules of one kind are together. Newly inserted rules since Ada 95 have been added rather haphazardly, which complicates understanding.

Ed S. wonders why the static expressions freeze at the end of a declaration list explicitly, doesn't that always happen at the end? Tucker explains that that occurs at the word **private** as well as **begin** or **end**. **Private** does not cause freezing other than in this case. This has been true since aspect specifications were defined, it is not new in this AI.

Jean-Pierre Rosen notes that Succ and Pred aren't included here. There doesn't seem to be any reason to omit them as they're also only dependent on position numbers. We add an AARM note to explain the model (the attributes are those that are static and well-defined without knowing the representation of the type).

There was a typo in one of the uses of aspect_definition (Tucker fixed it in the Google Docs version). We wonder why Google Docs didn't mark it as misspelled. It is suggested that the presence of the underscore is switching off Google's spell checking (no syntax terms ever seem to be marked as misspelled).

Approve AI with changes: 8-0-5. Ed S., Gary, Bob, Claire, Justin abstain.

### AI22-0048-1/01 Time_Zone parameter for Day_of_Week

There is a lengthy discussion of "Day_of_Week" vs. "Day_Of_Week". Tucker thinks that should be changed in the RM for consistency. The group mainly agrees to that.

Should we simply have a default here and not worry about the (compile-time) incompatibility?

The current definition of Day_of_Week appears to require using the local time, if we use a default of 0, that would change this to UTC. That would be a significant runtime incompatibility.

What does this do in existing implementations? Randy says Janus/Ada does local time. Richard says that some GNATs have problems in this area. Bob inspects the GNAT source code and says it is not obvious what it does because it calls other functions to get the answer.

Tucker will take this to find out what implementations do for Day_of_Week.

Keep alive: 14-0-0.

### AI22-0049-1/01 Seconds function with Time_Zone parameters

Richard suggests that the name be "Seconds_Of_Day". Then he suggests that we overload "Seconds_Of" with this function. Ed S. doesn't like the overloading of this one.

Bob suggests "Second_Of_Day". [I didn't record why he thinks the singular form is better – Editor.]

We briefly discuss using the name used in Calendar ("Seconds"). We agree that it is too close to the function "Second".

Straw poll on the names.

We ask if anyone objects to particular names.

> Seconds_Of: Object to: 0.
>
> Seconds_Of_Day: Object to: Richard, Bob.
>
> Second_Of_Day: Object to: Tucker, Randy, someone else.
>
> Seconds_Elapsed: Object to: Tucker.
>
> Seconds_At: Object to: Tucker.

Intent is to go with Seconds_Of, since no one objected to it formally.

Randy asks Ed S. about his earlier complaint about having additional overloading. He still has it, but of course overloading commonly exists in Ada. Richard notes that this package is full of overloading: 4 Time_Ofs, 5 Splits, etc.

Richard will take the AI and update the name and discussion.

Approve intent: 12-0-1. Steve abstains.

### AI22-0051-1/02 Preelaboratable_Initialization and contract aspects

Ed Fish enumerates the possible ways that this can be solved. Randy was confused by the penultimate item; Ed meant it to only apply in preelaborable contexts (that is, the declaration of library-level default initialized objects in preelaborable packages).

We go through the list. Allowing execution brings up a number of problems that preelaboration is intended to prevent. And existing implementations may depend on the fact that execution can't happen. This does not seem to be a good choice.

Analysis of the assertions would be a new thing. You'd have to look in bodies of other units to do the analysis. We'd also have to define precisely what analysis is required (so portable code is possible), that could be quite a few additional rules.

Disallowing P_I with D_I_C, either allowing simple expressions or not would be problematic for the containers and other ADTs. For any private type, one has to call at least one function in a D_I_C or other contract, and that function cannot be made static (there are no static private types). One could expand the definition of static to include "user-defined" static types, but that is fairly complex (it was tried in an early version of AI12-0175-1), and it still would be a fairly limiting approach.

Ignoring contracts in preelaborable contexts was the suggestion that got the most traction last time. Tucker thinks this can be considered a special assertion policy that causes contracts not to be evaluated for library-level default initialization.

Deferring evaluation of the check to later (probably after all preelaboration is complete) would be fairly complex to implement. We should allow the check to be made later, but we would not want to require that.

Steve asks Claire if there is any SPARK impact; she says that they don't care, as the check will be made at compile-time regardless of which rule is applied for Ada purposes.

Approve intent: 14-0-0.

Ed Fish will take the AI to provide wording with Tucker's help.

### AI22-0052-1/01 Storage Pool-related side effects

Tucker tries to explain the solution.

Randy wonders if we need this; there is an implicit call to Allocate associated with an allocator. That should carry any storage pool side-effects associated with pool object. Why do we need this separately?

We decide to analyze this further, as it may not be necessary at all. Steve takes the AI to analyze the reasons for it. We decide not to take any vote.

### AI22-0053-1/02 An unintended consequence of AI12-0101-1

Randy forgot to put this one into Tucker's homework list, and he didn't edit the wording. Tucker doesn't remember what his concern was. We check the last meeting's minutes, but they only say that Tucker doesn't like the wording.

We give it back to him to figure out his concern and then wordsmith.

### AI22-0054-1/01 Ambiguous prefix for the Result attributes

Steve discusses the AI. We already have a case where the context can affect the resolution of a prefix (‘Access), so having another does not seem too bad. Tucker noted that the Caller attribute is a similar case, so it was included in the wording.

The AARM note is new, it should be marked as an insertion.

Approve AI with changes: 15-0-0.

### AI22-0055-1/01 Usage Advice

Randy explains the issue. The RM traditionally has placed usage information into notes, where it is mixed in with other information about the language design. However, we are not allowed to use wording that suggests requirements or recommendations in notes. We were able to reword the requirement notes without losing too much of the point, but usage recommendations are impossible to reword without losing the point. (ISO rejected one of our attempts as a “covert recommendation”!) For the FDIS, we simply deleted the notes (leaving them in the Ada 2022 RM), but we need to choose a long-term path.

Randy noted four alternatives:

- A new part (separate document). While this is the Directives recommendation, it would put the advice far away from the definition that most are reading. And there is no energy to construct such a part.

- A new category subheading (such as “Usage Advice”). We probably should gather this advice in an Annex as well (similar to what we do for Implementation Advice). This is Randy’s recommendation.

- Just a new Annex containing all of the notes. This would put the advice far away from the definition again; who would remember to look in the Annex when reading about random numbers?

- Move the notes to AARM notes. This wouldn’t require any rewording, but it would not let causal readers of the RM see the advice. And most such advice is aimed at relative newbies to Ada, not the sort of people that would read the AARM.

Ed S. comments that usage information is mostly the domain of John and other authors. Randy points out that these notes go back to Ada 83; they have been considered important in each revision of Ada. It’s unusual to decide that we no longer need them after 40 years.

The group agrees with the idea of a new category header and annex. We turn to the name of the header.

Tucker thinks that a broader category is better. Tucker suggests that “Usage” be the header, and that would include most if not all examples. [Editor’s note: While that would work well with the FDIS format, we don’t use that format in the RM – in the RM, “Examples” are a category header. Reformatting the RM would be needed.]

Tucker suggests doing this for a few sections on a trial basis, and then see how it looks. This sounds like such a good idea we assign it to him.

Keep alive: 14-0-0.

### AI22-0058-1/01 Preconditions for checking Task_Ids

Tucker notes that we decided to use **not** and **or else** in preconditions, not conditional expressions.

Tucker is concerned that the precondition does not explicitly show the part of the precondition about Null_Task_Id. This is somewhat of a philosophical issue – should the entire precondition be written locally, or is it OK to depend in

part on the preconditions of the functions called by the precondition? We don't really come to an answer on this question (some of us believe that the simplification adds readability without sacrificing correctness, OTOH some tools consider it wrong). But we look at what the precondition would look like with all of the checks written explicitly.

Randy notes that there is no use clause for Ada.Task_Identification, so we would have to use the full expanded name for /= and Null_Task_Id. Tucker prefers to give the use clause, giving:

```
(declare
    use Ada.Task_Identification;
 begin
    (T /= Null_Task_Id or else raise Program_Error)
       and then
    (not Is_Terminated (T) or else raise Tasking_Error));
```

We could just add the use clause to the context clause of the packages. That seems better; it is unlikely to conflict with anything. Randy wonders if we should simplify the rest of the package in that case (the parameters use the fully expanded name for Task_Id and Null_Task_Id). Bob comments that they can't believe that Mr. **Use** Adverse would say something like that, and the conversation devolves (and the question never is answered). [Randy would like to add that consistency trumps philosophy; he would have written a function for these preconditions given in Ada.Task_Identification (the original proposal) and called it using a full expanded name. No **use** clauses anywhere, nor needed. But if we're going to use a **use** clause, then we should use it consistently.]

```
  with Pre => (T /= Null_Task_Id or else raise Program_Error)
                  and then
              (not Is_Terminated (T) or else raise Tasking_Error));
```

Approve AI with changes: 15-0-0.