Notes on telephone ARG meeting (#43B), April 7, 2011 1:00 PM EDT

Attendees: Steve Baird, John Barnes, Randy Brukardt, Gary Dismukes, Bob Duff, Jean-Pierre Rosen, Ed Schonberg, Tucker Taft.

Summary of Action Items (finish by April 21, 2011):

Tucker:

- AI05-0115-1 Redo wording to allow conversions (explicit or implicit) so long as enough information is visible, characteristics are based on the parent's view.
- AI05-0183-1 Determine why Address is view-specific, and justify or remove.
- AI05-0246-1 Write a sentence to describe the purpose of the restriction before launching into the details.
- AI05-0247-1 Work with Randy to figure out appropriate wording for this AI.

Randy:

- AI05-0247-1 Work with Tucker to figure out appropriate wording for this AI.

Bob:

- AI05-0153-3 Reword to eliminate bad term "predicate clause".
- AI05-0190-1 Use "Master of a function call" wording.
- AI05-0234-1 Define "Master of a function call" appropriately and use it.

Ed:

- AI05-0110-1 Reword the new text so that "including predefined operators" is clearly noting that these are included in "primitive subprograms" without being confusing.

Steve:

- AI05-0248-1 Propose a rewording of the Assign routines for the containers (there are 6) that does not use "Corresponding Elements". (This probably will be placed in a new AI.)

Jean-Pierre:

- AI05-0247-1 Make a proposal to require explicit syntax when "weakening" preconditions.

Gary:

- AI05-0110-1 Decide if we need a 7.3.1 cross-reference in 12.5.1(20/2) and possibility 7.3(16/2).

The following AIs were Approved (sometimes with changes):

- AI05-0110-1/03 Inheritance of characteristics of generic formal derived types (8-0-0)
- AI05-0215-1/06 Pragma Implemented should be an aspect, renames, and better defined (6-0-2)
- AI05-0229-1/08 Specifiable aspects (8-0-0)

- AI05-0243-1/04 Clarification of categorizations (7-0-1)
- AI05-0245-1/03 Introduction update for Ada 2012 (8-0-0)
- AI05-0246-1/01 Restriction No_Implementation_Identifiers and Profile No_Implementation_Extensions (8-0-0)

Start: 1:05 PM EDT

AI05-0245-1/03 Introduction update for Ada 2012

John sent a small suggestion in e-mail. We'll use it.

Ed suggests changes to the last paragraph. He doesn't like "primarily". Steve prefers the "primarily"; we'll leave it.

Tucker would like "some features" in that bullet. Better: "Certain features are added primarily..."

Approve with changes: 8-0-0.

AI05-0153-3/07 Subtype predicates

Randy had noted that the term "predicate clause" is wrong, because these are specified in aspect_specifications; they're not even allowed in an aspect_clause! Tucker had suggested changing the term to "predicate specification".

Gary suggests dropping the term altogether, and just using "predicate aspect" everywhere.

Bob will take an action item to decide which works better, and will rewrite the wording in the appropriate way.

This is already approved and doesn't need a new vote.

AI05-0183-1/11 Aspect specifications

The AARM note about the types of declarations on which aspects are allowed is wrong, Randy says that he updated it in the standard, but not the AI.

Ed wonders why Address is specifically mentioned as "view-specific". No one knows, Tucker will take this question off-line and explain or recommend to remove.

Randy notes that he added the following (13.3.2(32/3) in his working draft) to make sure that visibility applies to aspects when that makes sense:

"If a Legality Rule or Static Semantics rule only applies when a particular aspect has been specified, the aspect is considered to have been specified only when the aspect_specification or attribute_definition_clause is visible (see 8.3) at the point of the application of the rule."

The wording is icky. Tucker thinks something closer to what "availability" defines is better. [After meeting note: "Availability" doesn't have to worry about apply to rules that it doesn't know about. Randy considered defining a special term for this case, but one is not obvious and it would require rewording elsewhere. But doing so would have the advantage of not being close to the 13.1 meaning of "specified" (which might be different).] Tucker will try to wordsmith this.

Delete the last part of the AARM Discussion of the Implementation Permission. "at least in this version." This was another thing done in the standard draft and not in the AI.

Vote this by e-mail ballot after Tucker's update is finished.

AI05-0247-1/04 Preconditions, Postconditions, multiple inheritance, and dispatching

Randy tries to explain his suggestion for requiring overriding. We discuss whether we need that for class-wide postconditions.

Postconditions belong to the body, but when they are inherited, we want to use the ones associated with the call. Tucker would like to somehow describe that the "inherited body" has new postconditions. Randy suggests 3.9.2(2x) might cause problems with the model (it says that the body invoked is the one of the parent). Tucker says that the old wording is just plain wrong.

We discuss what we want:

All postconditions and type_invariants apply to an inherited routine. These will not require overriding.

If a routine has more than one direct ancestor, the routine requires overriding unless all of the inherited class-wide preconditions are the same. (The preconditions are the same if all of the class-wide preconditions are "True" or not specified, which means that they are "True"). [This will necessarily violate LSP.]

A renames used as an overriding that tries to change the class-wide precondition is illegal (for the same reason as the above). A user can always write a body containing a call in both of these cases if they don't care that they are violating LSP – that shows that they intended to do so.

Call through access-to-subprogram is a call through a wrapper (determined by the point of 'Access).

Randy and Tucker will try again to word this.

Back to preconditions: Jean-Pierre says that he would like to see a syntactic gizmo to say that a Pre'Class is "or"ed to an existing preconditions. "Pre'Class => [else] expression". Tucker and Ed don't like it. Randy supports Jean-Pierre, noting that Eiffel thought this was important (it does this).

Tucker suggests that you have a way to write the original one: "Inherited'Pre or <expression>". But he doesn't want to require this form, it is too much noise.

Tucker and Ed don't like to indicate inheritance. Ed notes that "overriding" on the subprogram will indicate that.

Bob says he sees both sides, and he would like to see a concrete proposal. He tries to ask whether this gizmo is required any time that there is "or"ing, such as when an interface and a concrete routine come together. But no one answers him [the editor thinks that's a good idea; it's the same case that we're requiring overriding].

Jean-Pierre says that Ada is about avoiding error, and remembering that there is an inherited precondition is a huge source of error.

Randy notes that he thinks that changing an inherited class-wide precondition will be rare; usually the inherited one is exactly right (since something stronger is a bad idea). So it will be unusual to see these in code, and remembering the special semantics in that case will be hard to do.

Jean-Pierre is given an action item to make one or more proposals on this topic.

Tucker would like to make this AI a complete replacement of the old wording, so it becomes easier to read and hopefully understand.

Approve intent: 8-0-0.

AI05-0234-1/05 Hole in AI05-0051-1
AI05-0190-1/10 Global storage pool controls

These two items are discussed together. Bob says that the definition in AI05-0234-1 doesn't work really well for AI05-0190-1.

He worries that this is defining an accessibility level, he would rather that it defines a "point of call", and then that defines accessibility level.

Tucker suggests that we say that a "master" is determined by the point of call, rather than an "accessibility level". Then this bullet would say that the collection is in that master.

Bob still prefers "ultimate master"; Tucker does not like "ultimate" because it seems to promise too much. We decide to use "*Master of the result of a function call* is determined as follows", then change existing uses of "point of the call" to use this master.

So we think that we should make these changes.

Bob does not want this action item, but he gets it anyway since he is the one that wants the change.

He also will update AI05-0190-1 in the same way.

John says in AI05-0190-1 "is a storage pool is pool" needs to say something else.

AI05-0229-1/08 Specifiable aspects

Randy has a list of questions about this wording.

Atomic_Components, Volatile_Components, Independent_Components: the type-only model is preferred, so change to use that. That will take a bit of additional wording, but the model is cleaner.

Preelaborable_Initialization:

Tucker suggests "known to have preelaborable initialization", in place of "eligible". But that conflicts with "have preelaborable initialization".

Tucker thinks that "have preelaborable initialization" in 10.2.1(9/2) should be "known to have preelaborable initialization".

There is legality rule missing here, the aspect cannot be specified for a full view if there is a partial view.

Tucker says that the default value rule is view-specific. In fact the current rule, makes full views not have preelaborable initialization if the partial view does not have it. That would be incompatible.

Randy agrees that this is the case, he then suggests abandoning this as an aspect, it doesn't fit it as a model. We could try to fix this up (Steve volunteers to try), but this is not a critical change, and it is getting late for such heroics. We decide to abandon this change and leave this as a pragma.

Yes, we need to disallow Storage_Size, Priority, et. al. on interfaces. Randy will write a legality rule to do that. Someone thinks this might be a useful capability. Perhaps, but we aren't trying to add new possibilities in this AI – and what happens if there are different priorities in two progenitors of a task type??

Should convention pragmas in predefined packages be replaced with aspects? Steve says if there was no work, we should do it. After some discussion, we decide that they should be changed.

Tucker asks that others reread B.1, since it is nearly totally rewritten.

Approve AI with changes: 8-0-0.

AI05-0243-1/04 Clarification of categorizations

In an AARM note: change OTOH should be However.

Randy notes that he would really prefer a single categorization aspect. But it would be (slightly) incompatible, and wouldn't handle Preelaborate. So we decide to leave this as is.

The difference between a library_item and a library unit is explored; the AI wording is correct.

Approve AI with change: 7-0-1.

AI05-0215-1/06 Pragma Implemented should be an aspect, renamed, and better defined

John e-mailed some typos.

Approve AI with changes: 6-0-2.

AI05-0246-1/01 Restriction No_Implementation_Identifiers and Profile No_Implementation_Extensions

John wonders what this random collection of identifiers is about. Randy explains that these are all of the places where an implementation can add implementation-defined stuff in language defined packages; there are no others.

It would be good if the description said that. It should start with that.

There is a suggestion that it *only* say that. But that makes the restriction scary, users will have no way to guess if the restriction might be a problem or not. Bob says that they should just add it and let the compiler tell them. Not everyone works that way; it's hard to imagine someone doing that in a multi-million line project. So we will leave the complete list here.

Tucker will write a new sentence to introduce this restriction.

Approve AI with changes: 8-0-0.

AI05-0248-1/01 More rewordings from the first editorial review

Since these were just posted last night, we'll have a later letter ballot on this AI.

Randy asks about the open issues.

Gary complained about "corresponding elements" in the Assign. That seems wrong for most of the containers. Steve will try to reword all of the Assign routines.

Gary will look through the standard to find missing commas where adding them cannot be construed to change the meaning. Randy will then change them without change markers or putting them into the Amendment (as we did for hyphens in mode-conformant). Steve and John agree that would be a good idea.

Locale: Should we change the names of the constants? Tucker says "do it". Approve: 7-0-1.

H.4(24): Bob says this isn't important enough to fix. Someone suggests that it is more important than the commas. We'll do this after Ada 2012.

Bob had suggested rearranging Chapter 13. But it is too hard to rearrange. Tucker suggests changing the title. Bob suggests "Hodgepodge of stuff". Tucker suggests "Operational and Representation Control". John says "Others" [The editor thinks this was a joke. He hopes.]

Tucker wants a naming contest, for one week, to come up with a name. Prize is a bottle of scotch at Edinburgh. (He didn't specify who was buying.)

AI05-0115-1/07 Aggregates with components that are not visible

Tucker notes that we don't want to inherit characteristics from a parent. But it is not a clear that this is a good thing for type conversion. Steve asks if there would be cases where you can convert A to B and B to C, but not A to C.

Tucker will try to come up with wording that has that effect.

Approve intent: 6-0-2.

AI05-0110-1/03 Inheritance of characteristics of generic formal derived types

Ed does not like "(including predefined operators)" because the wording just before the text also uses "include". The "including" here applies to "primitive subprograms". Ed will try to reword this.

Gary wonders why the 7.3.1 reference was dropped in 12.5.1(20). It's not clear whether we need this reference. Gary will decide whether we need this cross-reference.

John says there is a typo in discussion: "be have".

Approve AI with changes: 8-0-0.

Stop: 4:15 PM