ARG Rapporteur's Proposal for Defining Scope of Amendment 2 to ISO/IEC 8652:1995
ISO/IEC JTC1/SC22/WG9 N 505
June 4, 2010

This scope document is based on the instructions to the ARG found in WG 9 document N498.

Excerpts from document N498 appear below in **bold font**. My additional comments appear in regular font. I have added a few bulleted subcategories in cases where N498 only gave broad advice and a few examples.

Because this is a scope document, rather than a detailed document, I only give AI numbers, without dates or versions. I only cover AIs that extend or significantly change the language here; there are also a significant number of AIs that fix errors in the language definition and thus must be included in any language update.

The AIs that are not marked with a symbol have been approved by WG9, or approved by the ARG and will go for approval to the next WG9 meetings.

The AIs that are still under discussion are marked with one of the following symbols after their number:

* This AI still needs some more work before it can be submitted to WG9, but its technical content is well-defined and believed to be sound (17 AIs).

& This AI still needs substantial work before it can go to WG9, and its technical content is still in a state of flux. It is still alive because the ARG sees sufficient value in the ideas being proposed, and therefore wants to study them some more before making a final decision. Note that in general there is no firm consensus on these AIs yet, as many people want to see the AI mature before forming an opinion. As such, some of these AIs may be dropped from the final Amendment (11 AIs).

The ARG is requested to pay particular attention to the following two categories of improvements:

A) Improvements that will maintain or improve Ada's advantages, especially in those user domains where safety and security are prime concerns;

B) Improvements that will remedy shortcomings in Ada.

Improvements of special interest in these categories are:

- **Improving the use and functionality of the predefined containers;**

AI05-0001-1 Bounded containers and other container issues
AI05-0069-1 Holder container
AI05-0136-1 Multiway tree container
*AI05-0159-1 Queue containers

These AIs add additional kinds of containers and helpers for containers. These are all new language-defined packages.

&AI05-0135-2 "Integrated" nested packages
*AI05-0139-2 Syntactic sugar for accessors, containers, and iterators
*AI05-0142-4 Explicitly aliased parameters
AI05-0143-1 In Out parameters for functions
*AI05-0144-2 Detecting dangerous order dependences
*AI05-0212-1 Accessors and Iterators for Ada.Containers

These AIs improve the usability of the containers in various ways. AI05-0139-2 provides convenient ways to access elements in containers and write iterators on containers. AI05-0142-4 provides static checking for safety when accessing elements in containers. AI05-0143-1 allows modification of elements via accessors (as well as eliminating a long-standing complaint about functions in Ada); AI05-0144-2 improves the safety of AI05-0143-1 by making some non-portable calls and attendant side-effects illegal. AI05-135-2 provides visibility improvements which make it easier to use containers in specifications that declare private types. AI05-0212-1 modifies the existing as well the new container packages to use all of these new features.

AI05-0184-1 Compatibility of streaming of containers

This AI better defines the meaning of container streaming, so that the container kind (bounded, unbounded, indefinite) does not change the streaming behavior.

- **Improving the ability to write and enforce contracts for Ada entities (for instance, via preconditions);**

AI05-0145-2 Pre- and Postconditions
*AI05-0146-1 Type and Package Invariants
*AI05-0153-1 Subtype predicates
*AI05-0183-1 Aspect Specifications
&AI05-0186-1 Global-in and global-out annotations

These AIs introduce various new kinds of contracts to Ada entities , and a new unified syntax for specifying them.
Note : AI05-0146-1 is now restricted to types, even though packages are still mentioned in its title.
Note: AI05-0186-1 is very likely going to be dropped.

*AI05-0147-1 Conditional expressions
*AI05-0158-1 Generalizing membership tests
AI05-0176-1 Quantified expressions
&AI05-0177-1 Parametrized expressions
*AI05-0188-1 Case expressions
&AI05-0191-1 Aliasing predicates

These AIs expand the expression capabilities of Ada to make it easier to write and analyze contracts. Of course, they can be used in many other contexts as well, and add to the expressiveness of the language in general.

- **Improving the capabilities of Ada on multi-core and multi-threaded architectures;**

&AI05-0117-1 Memory barriers and Volatile objects
*AI05-0167-1 Managing affinities for programs executing on multiprocessor platforms
AI05-0169-1 Defining group budgets for multiprocessor platforms
*AI05-0171-1 Ravenscar Profile for Multiprocessor Systems

These AIs were developed and recommended by IRTAW.

- **Improving the safety, use, and functionality of access types and dynamic storage management.**

&AI05-0111-1 Specifying a pool on an allocator

This AI provides a basis for region-based storage management in Ada.

AI05-0148-1 Accessibility of anonymous access stand-alone objects
AI05-0149-1 Access types conversion and membership
AI05-0152-1 Restriction No_Anonymous_Allocators

These AIs make use of anonymous access types safer. AI05-0148-1 makes anonymous access stand-alone objects much more useful by eliminating accessibility failures from them. AI05-0149-1 makes it possible for users to avoid accessibility failures with explicit tests (this will also be useful in contracts). AI05-0152-1 allows users to prevent the creation of allocators with unknown lifetimes .

AI05-0151-1 Allow incomplete types as parameter and result types

AI05-0162-1 Allow incomplete types to be completed by partial views
&AI05-0213-1 Formal incomplete types

These AIs expand the use of incomplete types; all of them get extensive use in the new and updated containers packages (and could have been listed in connection with them). AI05-0151-1 makes use of limited_with clauses within package specifications easier by reducing the need for access types (of all kinds). AI05-0162 eliminates a pointless restriction from incomplete types. AI05-0213-1 allows incomplete types to be used as generic formal parameters.

AI05-0189-1 Restriction No_Allocators_After_Elaboration
AI05-0190-1 Global storage pool controls
AI05-0193-1 Alignment of allocators

These AIs all help user-defined storage pools be more effective. AI05-0189-1 helps long-lived applications avoid heap use that could cause resource leakage via fragmentation. AI05-0190-1 helps applications use user-defined storage pool as the default storage pool. AI05-0193-1 gives user-defined storage pools more flexibility with alignment.

Additional improvements in category A (improve Ada's advantages):

AI05-0123-1 Composability of equality

This AI improves the reusability of Ada abstractions by making user-defined equality participate in predefined equality for all record types (not just tagged types).

&AI05-0127-1 Adding Locale Capabilities

This AI adds a locale query to Ada to make it easier to create programs that operate in multiple natural languages.

*AI05-0137-2 String encoding packages
*AI05-0185-1 Wide_Character and Wide_Wide_Character classification and folding

These AIs add packages that improve the handling of Unicode text.

AI05-0166-1 Yield for non-preemptive dispatching
AI05-0168-1 Extended suspension objects
*AI05-0170-1 Monitoring the time spent in Interrupt Handlers
*AI05-0174-1 Implement Task barriers in Ada

These AIs represent real-time improvements. These have also been developed and recommended by IRTAW.

Additional improvements in category B (remedy shortcomings in Ada):

AI05-0003-1 Qualified expressions and names
AI05-0015-1 Constant return objects
AI05-0030-2 Requeue on synchronized interfaces
AI05-0032-1 Extended return statements for class-wide functions

AI05-0003-1 allows qualified expressions to be treated as names, making prefix notation calls more useful. AI05-0015-1 and AI05-0032-1 remedy limitations in the extended return feature added to Ada 2005. AI05-0030-2 adds the capability to perform requeues on interface operations, making operations of synchronized interfaces more consistent with regular entries.

AI05-0031-1 Add a From parameter to Find_Token
AI05-0049-1 Extend file name processing in Ada.Directories

These AIs correct various small deficiencies in the Ada predefined packages.

&AI05-0119-1 Package Calendar, Daylight Savings Time, and UTC_Offset

This AI fixes problems using UTC times in Ada programs.

&AI05-0125-1 Nonoverridable operations of an ancestor

This AI eases structuring issues with OOP inheritance.

AI05-0150-1 Use all type clause

This AI provides use-visibility for enumeration literals and primitive operations, without the perceiveddisadvantages of the package use clause.

&AI05-0154-1 Unconstrained 'Access on array components

This AI eliminates an annoyance with no sane workaround. (But it's not clear that the solution is worth pursuing).

AI05-0161-1 Restrictions for default stream attributes of elementary types

This AI allows systems that depend on redefined stream attributes to get errors if the predefined ones are used by mistake.

AI05-0173-1 Testing if tags represent abstract types

This AI allows testing to prevent an exception from being raised by tag and stream operations.

AI05-0179-1 Labels at end of a sequence_of_statements

This AI allows more liberal placement of labels (especially useful as the equivalent of "continue").