Report of Session: Current Real-Time AIs

Chair: Michael Gonzalez-Harbour
Rapporteur: Tullio Vardanega

1 Introduction

The session chair recalls that the session goal is to recap the situation with the existing AIs that directly involve the IRTAW group and to instigate action this week on those that need further work.

AI	*Title*	*Status*
00249	Ravenscar Profile	Approved (WG9)
00250	Extensible Protected Types	Dormant
00265	Partition Elaboration Policy	Approved (WG9)
00297	Timing Events	Approved (ARG)
00298	Non-Preemptive Dispatching	Approved (ARG)
00307	Execution Time Clocks	Approved (ARG)
00321	Definition of Dispatching Policies	Approved (ARG)
00327	Dynamic Ceiling Priorities	Work item
10266	Task Termination Procedure	Work item

The chair proposes to start discussion from a brief recap of any of the approved AIs on which there may be comments from the floor and then to move on to discuss on AIs 00326 and 10266, which are outstanding work items for IRTAW.

AI-00250 Extensible Protected Types : the group will not attempt to resurrect the AI as is; rather, it will place a discussion item in session "Generating new AIs" on possible support for a new AI on task and protected interfaces as outlined by Tucker Taft in: [Ada User Journal 24-2, 125-127].

AI-00297 Timing Events : Michael Gonzalez-Harbour points out that his group's prototype implementation of this feature shows that the current AI is designed for very effective execution on bare runtimes while it may be slightly under-performing on top of POSIX operating systems.

AI-00298 Non-Preemptive Dispatching : Alan Burns comments that this AI may be the first of potentially many additional dispatching policies for getting flexible scheduling in Ada. Alan also notes that non-preemptive dispatching (NPD) enables the implementation of a non-dispatching version of the Ravenscar Profile, which the group continues to view as a *new* profile instead of just plain Ravenscar with NPD as argument.

AI-00307 Execution Time Clocks : Michael Gonzalez-Harbour reports on his group's, library-package based, implementation of this feature and on the problem of managing multiple execution-time timers from a single task (cf. the paper on this very subject in these proceedings). The difficulty stems from the exclusion of a "delay until CPU-time" construct from the AI, which limits one to call the single entry of the sole Timer object allowed for any given task.

Michael's paper discusses two possible additions to the current AI: (1): a Set_of_Timers protected type, so that a task can queue itself on any number of Timers, each identified by an out reference parameter; (2): a Cancel procedure in the protected Timer.

Brian Dobbing suggests a very simple and effective design pattern for multi-way timer entry: a bit array (1..Max) on whose bitwise false condition the caller waits and a Set_Bit procedure to be called by the expired timer.

Andy Wellings suggests another alternative that separates synchronisation from arming:

```
protected type Timer_Expired is
  entry Wait (T : out Task_ID);
  procedure Announce(T : Task_ID);
end;
type Expire is access Timer_Expired;
-- the Timer protected type is passed
-- an Expire value as a discriminant
protected type Timer
  (T : access Ada.Task_Identification.Task_ID
   E : Expire) is
   -- as in current AI
end;
```

This approach removes the Wait entry from the Timer protected type, so that the timer expiration will directly invoke the procedure (including null) that the user wants to treat the event.

The group finds Andy's approach attractive and invites Michael and Andy to work on a new write-up of

the AI to be presented and discussed at the "Generating New AIs" session later in the week.

2 Discussion of in-progress AIs

AI-00327 Dynamic Ceiling Priorities

Jorge Real, the initiator of this AI recalls the initial version of it, which introduced two protected procedure attributes: 'Set_Ceiling(P : priority) and 'Get_Ceiling(P: out priority). Jorge also notes that using 'Get_Ceiling for polling the actual uptake of a new-mode ceiling value is a poor programming style that may lead to program error (ceiling violation).

Javier Miranda provides a short outline of the prototype implementation of the AI in its initial form, based on GNAT v3.15p, as discussed in paper "Dynamic Ceiling Priorities in GNAT Implementation Report" included in these proceedings. The good news is that the implementation incurred very low cost in terms of compiler and runtime adaptations. The presence of pragma Adjustable_Ceiling, foreseen in the initial AI, proved superfluous and of no use in saving implementation cost. Basically, what the implementation has to do is to add and manipulate a new field New_Ceiling to the record that represents the PO protection level.

The issue of whether the priority of the calling task should be updated as part of setting the new ceiling value is then discussed. Obviously this decision also depends on when the new ceiling actually takes effect, which even in the latest formulation of the AI, is required not to occur until the end of the protected procedure. This notwithstanding, the group's view was that the epilogue code of the calling task should be executed at the old ceiling value, so as to preserve the conditions under which the calling task seized the protected objects. This view descends from the understanding that the caller's priority must be right with respect to the ceiling only at the time of the call, and not necessarily throughout the call.

A corollary of this issue concerns implementation of this AI feature on POSIX, in which the standard does not specify whether a thread can attempt a set a new ceiling (by calling pthread_mutex_setprioceiling()) on a lock it already owns. Michael Gonzalez-Harbour will prepare and send an interpretation request to the POSIX standardisation group on this issue.

The discussion then moves to the later version of the AI, in which the read/write 'Priority attribute replaces the two protected procedure attributes discussed above. Clause 9.5.1(2) of the ARM has an impact on the use of this feature, as it requires that only protected entries or procedures may write the 'Priority attribute.

The slightly different shape of the AI does not change the group's view that the record structure that represents the PO protection level must hold both the old and the new value of the ceiling, until the change takes effect externally.

An issue arises with the interpretation of the current AI where it states that it is a bounded error to lower the ceiling priority while there are tasks queued on an entry with higher priorities. The issue regards whether the check for this error should be made at the point of ceiling change. Brian Dobbing addresses this issue by recalling that the "eggshell model" implementation of protected objects in Ada 95, requires a one-time ceiling check for tasks that have entered the shell. Brian notes that retaining this model in the AI (which is the right thing to do anyway) would also reduce the magnitude of bounded error situations only to the case of tasks that find themselves enqueued on a closed barrier after a ceiling change, because at that point in time their active priority may no longer be compatible with the ceiling in effect. This erroneous situation, however, is so limited in scope that it can be resolved programmatically.

Andy Wellings notes that the use of 'Priority for the new ceiling value in a guard under the eggshell model combined with the proxy model (which defers the effect of the ceiling change until the end of the protected action) causes the guard to stay closed and the tasks held behind it to be released only upon the subsequent evaluation of the guard. The group expresses the view that the use of 'Priority in guards should then be disallowed in much the same way as ARM C.7.1(14) does not allow 'Caller to be used in guards.

At this point, the group contrasts the old and the new formulation of the AI and views the latter as less convincing.

The group asks Jorge Real and Brian Dobbing to prepare a new write-up for the AI reflecting on the proceedings of the discussion, and to present it at the "Generating New AIs" session later in the week.

AI-10266 Task Termination Procedure

Alan Burns recalls, for the benefit of the group, the intent served by the AI. Brian Dobbing enquires whether a set termination handler may ever happen to become an out-of-scope procedure. Alan replies that this should not be possible because it is always the master task that sets the default handler. Still, an issue exists with the downward closure of scopes.

The group does not view any point in discussing this AI any further, only noting that it is likely that the feature be more useful under the Ravenscar Profile than under the whole language.