Session: Language Issues

Chair: Tullio Vardanega
Rapporteur: José F. Ruiz

1. Introduction

This session focused on discussing open issues that arose from consideration of the real-time part of the Ada 2005 LRM. Three issues were table for the workshop to discuss. Accordingly, the goals of the session were to:

- Agree on proposed correction to EDF semantics
- Agree on resolution to requeue problem
- Review Ravenscar extensions for 2005 and consider extensions for distributed systems

2. Correcting the EDF definition in Ada 2005

The first part of this session addressed a problem in the current description of the Earliest Deadline First (EDF) dispatching protocol in the Ada 2005 standard. The difficult part in the definition of the EDF policy is the description of Ted Baker's Stack Resource Policy (SRP) for resource sharing [1].

The problematic wording is in the definition of the rules of EDF dispatching [3, D.2.6]. In that clause, the active priority of a task T when first activated or while it is blocked is defined as the maximum of the following:

- 24/2 the lowest priority in the range specified as *EDF_Across_Priorities* that includes the base priority of T,
- 25/2 the priorities, if any, currently inherited by T,
- 26/2 the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling priority P and task T has an earlier deadline than all such tasks.

Clause 26/2 contains the key semantics, and it is intended to ensure the behavior required by Baker's algorithm. Its current wording implies that a task could be placed on a ready queue above one on which a task with a shorter deadline is placed. Alan Burns described one scenario (the full details are in their position paper [2]) showing that the current wording is not in full accordance with Baker's protocol.

Alan Burns then illustrated a simple rewording to 26/2 which makes the rule stricter and achieves correct EDF dispatching in full adherence with Baker's SRP algorithm. The proposed rewording is as follows:

- 26/2* the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling priority P and task T has an earlier deadline than all such tasks and all other tasks on ready queues with priorities strictly less than P.

The workshop agreed that the current wording in 26/2 is not correct, approved the proposed reworking of it and tasked Alan Burns (as a member of the ARG) to produce an Ada Issue (AI) to illustrate the problem and propose the fix to it.

Michael González Harbour proposed to use a model checker to ensure that the definition complies with the intended semantics.

3. Requeuing via interfaces

The second part of the session discussed the possibility of allowing, in some form, requeuing via synchronized interfaces.

Timed and conditional entry calls, as well as asynchronous transfers of control (ATC), can use an interface as the target and the triggering event respectively, so it seems natural to allow requeue to an interface, both for completeness and consistency.

Andy Wellings explained that, as stated in their position paper [4], if requeue to an interface were allowed there are four cases whose semantics need to be defined:

- Requeue to an entry. This would follow normal requeue semantics.
- Requeue to a function (inside or outside a protected object). This would be an error condition that can be

caught at compile time, because there are no circumstances whereby a function in an interface can be implemented by an entry. This is similar to the illegal case where a function call is used as the target of a timed/conditional entry call or as a triggering event in a select-then-abort statement.

- Requeue to a protected procedure. If a protected procedure is used as the target of a timed/conditional entry call or as a triggering event in a select-then-abort statement then the protected procedure is executed immediately, as if it were an entry with a "when True" barrier. Semantics for requeuing would be the same.

- Requeue to a "regular" procedure. If a procedure is used as the target of a timed/conditional entry call or as a triggering event in a select-then-abort statement then the procedure is executed immediately. It would appear natural to do likewise when requeuing, but that would mean that the procedure would be executed at the ceiling priority of the original protected type (or priority of the original rendezvous), which is wrong.

The workshop agreed that requeue through synchronized interfaces is a useful and desirable primitive and should be supported in Ada so as to further the integration between object orientation and concurrency in Ada.

It was also agreed that the definition of requeue to an interface should be consistent with the use of interfaces in timed and conditional entry calls and asynchronous transfers of control. Consequently:

- Requeuing to an entry should follow normal requeue semantics.

- Requeuing to a function should be an error condition that can be caught at compile time.

- Two possible solutions exist to address requeues to both protected and regular procedures, which are discussed below:

  - A static scheme in which procedures within an interface can be identified as being "implemented as an entry" (by means of either a pragma, an "entry" identifier, or any other allowable mechanism), and requeues would only be accepted to procedures marked in that manner (which can be statically detected at compile time). If this approach were adopted, the semantics of timed/conditional entry calls and the select-then-abort statement might need to be revisited to make the operations illegal on procedure calls.
  - A dynamic scheme in which these calls are detected at run time, and then the calling task/protected objects accept statement/entry code's body is "completed, finalized and left" (see [3, 9.5.4, par. 7]) before the procedure is called.

The workshop agreed that the preferred option would be the static scheme. If this option were not viable then the dynamic scheme could be considered, if the implementation (run-time) overhead was found to be reasonable.

Andy Wellings will ask Javier Miranda to develop a prototype implementation to evaluate the impact of the considered options.

4. Distributed systems with Ravenscar

The final part of the session discussed extensions to the Ravenscar profile to address high-integrity distributed systems. The use of the Distributed Systems Annex (DSA) for high-integrity systems is very appealing because it is very easy to use and static analysis can be performed among partitions.

DSA has not been widely used in high-integrity systems because it is not real-time and, in the Ada 95 version, it uses *Ada.Streams* (which is not recommended for high-integrity systems). However, there are DSA implementations compliant with the Ravenscar profile (PolyORB [6] can be configured that way), real-time DSA (such as RT-GLADE [7]), and the new Ada 2005 standard does not require *Ada.Streams*. Hence, there are good foundations to build on, but high-integrity systems would require an additional set of restrictions to guarantee the required degree of predictability, efficiency, and simplicity.

Santiago Urueña then presented the list of restrictions proposed in their position paper [5], categorized into a set of mandatory and optional restrictions. The set of mandatory restrictions would be made up by the following:

- No remote access types. This would allow the static creation of every required connection for each remote operation. Stephen Michell commented that this restriction will also avoid problems with the variable size of attribute *'External_Tag*. He also indicated that we should probably go further and forbid remote types (because they require Ada.Streams). Andy Wellings pointed out that it must be taken into account that restricting this will imply disallowing object-oriented programming in distributed systems altogether, which seems beyond what a Ravenscar-like profile should sanction.

- No concurrent remote calls. It would ensure that no remote operation can be called while processing a past invocation, thereby simplifying response time analysis analysis. Michael González Harbour pointed out that

research works exist which target distributed systems showing how to take into account request queues in the analysis, so this restriction would not be strictly needed. He indicated also that the RPC receivers could be made more visible so that they can be dimensioned by the user and included in the analysis.

- Coordinated elaboration of partitions. The distributed application would not start until all its partitions have been elaborated, thereby improving determinism.

A further set of optional restrictions was also proposed that would simplify the implementation and facilitate response time analysis: no synchronous communication, no variable size messages, and no remote nested calls.

The workshop felt that there is a need to address high-integrity real-time distributed systems, and Ada is very well placed for that (it would be an interesting topic for next IRTAW). The workshop encouraged people to work on this topic in order to be able to define the list of requirements for such systems and the model to support them.

5 Summary

The following summarizes the positions taken by the workshop during this session:

- An Ada Issue (AI) should be produced to fix the definition of the EDF protocol.
- Requeue through synchronized interfaces should be supported in Ada. A static and a dynamic scheme were proposed (consistent with the use of interfaces in timed and conditional entry calls and asynchronous transfers of control) that need to be evaluated.
- There is a need to investigate models to support high-integrity real-time distributed systems.

References

[1] Baker, T.P. *Stack-based scheduling of realtime processes.* In Real-Time Systems, 3(1), March 1991.

[2] Zerzelidis, A., Burns, A., Wellings, A.J. *Correcting the EDF protocol in Ada 2005.* In Ada-Letters (this issue).

[3] Taft, S.T., Duff, R.A., Brukardt, R.L., Plöedereder, E., Leroy, P., eds.: *Ada 2005 Reference Manual. Language and Standard Libraries.* International Standard ISO/IEC 8652/1995(E) with Technical Corrigendum 1 and Amendment 1. Number 4348 in Lecture Notes in Computer Science. Springer-Verlag (2006).

[4] Wellings, A.J., Burns, A. *Integrating OOP and Tasking - The missing requeue.* In Ada-Letters (this issue)

[5] Urueña, S., Zamorano, J. *Building High-Integrity Distributed System with Ravenscar Restrictions.* In Ada-Letters (this issue)

[6] Vergnaud, T., Hugues, J., Pautet, L., Kordon, F. *PolyORB: a Schizophrenic Middleware to Build Versatile Reliable Distributed Applications.* In Proceedings of the 9th International Conference on Reliable Software Techologies Ada-Europe 2004 (RST'04), volume LNCS 3063, pages 106-119, Palma de Mallorca, Spain, June 2004. Springer Verlag.

[7] López Campos, J., Gutiérrez, J.J., González Harbour, M. *The chance for Ada to support distribution and real-time in embedded systems.* In Proceedings of the 9th International Conference on Reliable Software Techologies Ada-Europe 2004 (RST'04), volume LNCS 3063, pages 91-105, Palma de Mallorca, Spain, June 2004. Springer Verlag.