# ISO/IEC JTC1/SC22/WG9 N 563

## Report to JTC 1/SC 22/WG 9 Ada IRTAW 2016

*7 June 2016*

*Stephen Michell*

The 2016 edition of the International Real Time Ada Workshop series was held in Benicassim, Spain April 11-13 2016, attended by participants from Canada, Italy, Norway, Portugal, Spain United Kingdom and United States of America. The workshop discussed topics of concern to the real time Ada community.

The workshop was subdivided into six topic areas:

1. Deadline floor protocol issues,
2. Parallel programming issues,
3. Language issues,
4. Experience,
5. Language profiles, and
6. Time-based language vulnerabilities.

Each topic area is discussed below.

Deadline Floor

This session examined issues associated with the deadline floor protocol as first proposed at IRTAW 2013, and issues identified at that workshop. The paper considered was Burns and Wellings, “*The deadline floor protocol and Ada*”.

The workshop noted that the proposals would mean changes to the current EDF in Ada, such as the addition of explicit types and objects to represent deadlines. The workshop preferred extending the current package Ada.Dispatching.EDF.

The workshop recommended the deprecation of the Stack Resource Protocol.

The workshop recommended the support of a deadline floor aspect for protected objects. It was believed that the current aspect Relative\_Deadline could be overloaded to support protected objects.

Parallel Programming

The Multicore/Parallel Processing session examined issues associated with the addition of syntax to Ada to effectively manage parallel computation on multicore processors. The papers considered were: Michell, Pinho, Moore, Taft, “*Constraints on the Use of Executors in Real-time Systems*”, and Taft, Moore, Pinho, Michell, “*Reduction of parallel computation in the parallel model for Ada*”.

The majority of the effort was in solidifying constraints on parallel processing in real time systems. All systems are concerned with the relative efficiency and correctness of parallel computation vs strictly sequential computation, but real time systems add dimensions of determinism timing, overruns, and scheduling. The workshop determined that

* Priority changes should not be permitted within parallelizable code;
* Timing events should not be called or serviced within parallelizable code;
* Issues around the use of Set\_CPU for tasks with within parallelizable code are complicated enough that a new Set\_CPU should be created that permits the allocation of a task to a single CPU and its tasklets to a set of CPU’s;
* Work stealing or parent stealing leads to code that cannot be statically analysed using current methods and tools
* Programmers should be able to specify the maximum number of allowed executors and of active executors for the execution of tasklets, but there should be no mechanism to name or control executors;
* Tasklet control, in the sense of allocating “chunks” to tasklets during parallel execution is essential; and
* At this time, lack of analysis of highly parallel systems means that they should not be used in hard real time partitions.

Discussions were held on the reduction proposals from the second paper. There was general support for the notion of syntax to specify and control parallelism in Ada (as opposed to strictly library-based solutions). There was also support for the programmer to control aspects in the “map” and “reduce” aspects of parallelization.

Language Issues

The goal of this session was to discuss and, if appropriate, generate Ada Interpretations for several language

related issues presented to the workshop:

* Extension of the Synchronous Task Control in order to allow the use of Suspension Objects by concurrent tasks.
* Inclusion of Synchronous Barriers in the Ravenscar profile.
* Addition of execution timer and group budget support for interrupt handlers.
* Issues on High-Integrity Dynamic Memory Management.

The issues of synchronous task control and synchronous barriers was passed to the workshop by the WG 9/Ada Rapporteur Group via the contribution Burns, A. and Wellings, A.J. “*Synchronos Task Control and Synchronous Barriers”* for recommendation to the ARG in processing Ada Interpretations associated with these issues.

The issue of synchronous task control was based on the expectation that multiple tasks co co-ordinate access to a suspension object. The workshop confirmed that only a single task can suspend upon a suspension object, and that it should be a bounded error to attempt to permit more than one task to suspend on a single object.

The workshop considered the inclusion of synchronous barriers in the definition of Ravenscar. While there was a general consensus that synchronous barriers may be useful in Ravenscar systems, it was decided that more industrial use of Ravenscar on multiprocessor system is required before they could be included.

The workshop discussed a proposal by Kristoffer Nyborg Gregertsen, “*Revising the Ada timers and group budgets to support execution time control for interrupt handling*”. The paper was concerned with potential overruns in a realtime system due to interrupts and system level events. The workshop agreed that library mechanisms could be employed to cure the problems raised, but language syntax solutions are superior. There was insufficient support for the proposals to recommend language changes to Ada at this time.

The discussions on High-Integrity Dynamic Memory Management originated from the paper Wellings, A.J., Cholpanov, V. and Burns, A. “*Implementing Safety-Critical Java Missions in Ada*”. During the discussions of dynamic memory allocation replicating usage in safety-critical Java, a significant issue in the allocation, management and deallocation of memory allocated to objects, such as accessed objects and task stacks were identified. There should be an AI raised on this topic.

Experience

The experience session considered Real, J., Sáez, S., Crespo, A., “Combined Scheduling of Time-Triggered Plans & Priority Scheduled Task Sets”. This paper proposed mechanisms to combine tasking-based scheduling and cyclic scheduling approaches. The approach was well received. It grants minimum release jitter for selected tasks (scheduled by a highest-priority time-triggered scheduler, driven by timing events), while it also supports the execution of priority-based tasks with less strict jitter requirements. Several programming patterns were also proposed, ranging from a simple time-triggered task to more complex subtask decompositions proposed for control systems. Discussion led to the suggestion of additional patterns, such as breaking a long-running TT task into segments, and other subtask decompositions.

The workshop provided suggestions for further research, such as exploring a Ravenscar implementation that would ease certification; or considering the integration of this approach in a more general real-time utilities framework, such as the ones considered in previous editions of the workshop.

Language Profiles

The “Profiles” session examined various profiles, official and unofficial, that are used in Ada. The goal was to determine the desirability of formalizing language profiles as was done with great success for the Ravenscar Tasking Profile. The main paper Garrido J., Lacruz B., Zamorano J., de La Puente J.A., “In Support of Extending the Ravenscar Profile”, supported extensions to Ravenscar. The workshop considered extensions to Ravenscar, as well as other profiles that could be created for specialized programming in Ada.

There was strong resistance to extending Ravenscar in ways that would reduce determinism, such as permitting multiple entries in a protected object or permitting multiple tasks to queue on the entry queue. There was recognition that there is a strong desire to add extensions to Ravenscar to simplify the programming effort, and that there will be vendor-specific profiles provided that essentially do this extension.

It was strongly felt that, if a profile is developed that includes “Ravenscar” in its name, that this profile should not remove any functionality of the “base” profile, and should be essentially the same as the base profile. For example, the addition of “earliest deadline first” scheduling to Ravenscar may not qualify as an extension. Such a rule should apply to any profiles developed and later extended.

Time Vulnerabilities

The “Time Vulnerabilities” session focussed on a paper submitted by Michell, S. “*Time Issues in Programs Vulnerabilities for Programming Languages and Systems*”.

WG 23 is amending TR 24772 “Guidance on avoiding programming language vulnerabilities”. As part of that process, WG 23 is identifying vulnerabilities that have not been previously captured, either by WG 23, or by other organizations such as CWE, CERT, or MISRA. While WG 23 identified some concurrency vulnerabilities in edition 2 of TR 24772, WG 23 had not considered any issues with the management or use of clocks and time in programs.

The document that initiated this session identified issues associated with the use of different clocks or time bases in a system; representation of time (including granularity and fixed word sizes); perceptions of the passage of time; missed deadlines due to timing errors; and time effects due to virtualization.

The workshop agreed that the issues identified are vulnerabilities that need documentation, and identified 3 high-level vulnerabilities to propose to WG 23. At its teleconference after the workshop, WG 23 decided to incorporate these vulnerabilities, but to make them all application vulnerabilities and not language-based vulnerabilities. The issue for WG 9 is that there will be no expectation of language-specific wording for these vulnerabilities.