

Status and Future of the Ravenscar Profile

Session Summary

Andy Wellings
Department of Computer Science
University of York, York, UK.

1 Introduction

The session was chaired by Alan Burns who indicated that there had been several position papers received addressing various aspects of the Ravenscar Profile (RP). Based on the material presented in these papers, he suggested that the session consider the following topics:

- Implementation and Application Experiences
- Completing the Definition
- Extending the Model

The goal of the first topic was mainly to receive general feedback from the implementation teams and application programmers. These had resulted in several issues being raised. The RP only addresses issues relating to tasking; it is silent on all other aspects of the Ada language. However, some had felt that for a Ravenscar implementation to be effective, there were other restrictions that were necessary. These issues are reported under the second topic. Finally, there are many tasking profiles that could be defined. However, there had been no position papers making proposals for new subsets. Instead, papers had identified extensions to the RP that would give more expressive power but at little extra cost. These were discussed under the third topic. Issues to do with extending the RP for Distributed Systems were not considered in this session.

2 Implementation and Application Experiences

There were two implementation teams present at the workshop. Jose Ruiz summarised the ORK experience. They had implemented the RP using the GNAT system as a base. He indicated that in general they had found few problems implementing the RP and that the GNUALLI had been very useful. However, GNARL was more difficult to adapt. The specific issues he raised included the support for interrupt management was very complex and had to be re-implemented to be Ravenscar compliant the implementation of Suspension Objects was redefined to be Ravenscar compliant the Atomic pragma was not guaranteed for elements wider than a machine word

the overall footprint is too large; this was because they were not able fully to customise GNARL so that it supported the RP and no other facilities potentially blocking operations during protected actions had caused some difficulties they had initially wanted to provide protections between task stacks but the usual approach to implementing protected entries made this expensive

Brian Dobbing reported the Aonix experience. They had found few problems implementing the RP, and currently they had sold a substantial number of copies of the Raven product world wide, including 4 which had been used in certified systems: two at 178B level A, one at 178B level B and another probably at 0055 SIL level 4. (For example, the Sikorsky Helicopter's Health Monitoring Level B system.) However, he cautioned that one of these was not using tasking.

Brian also indicated that due to the simplicity of the RP, Aonix had seen a reduction of footprint size from approximately 500K (for a full Ada run-time) to approximately 3.5K for a RP run-time on a power PC. Furthermore, modified PIWG tests showed between 5 and 16 times increase in speed when running on Raven rather than the full run-time. The simplicity had also meant that there had only been one bug report in 2.5 years.

From the application programmers' point of view, Brian indicated that those customers who had tried to re-engineer Ada 83 code to the RP had struggled. However, those who were producing new software with an appropriate design methodology (like HRT-HOOD) had no problems. Furthermore, programmers were using substantial numbers of tasks (although level A systems tended to be more conservative). Joyce Tokar supported this view indicating that they had 178B level B applications using full Ada (Ada 83) which made substantial use of tasking.

3 Completing the Definition

3.1 Exceptions

The RP is silent on whether exceptions should be supported in the sequential part of a Ravenscar-compliant system. In general, there is no consensus on the role of exceptions in high-integrity systems. Some felt that exceptions have no place, others felt that they are a key component. One potential compromise is that they can only be caught at the outer level of a task. Lars Asplund's position paper considers the added complexity that exceptions bring to formal modelling. Stephen Michell's paper proposed that the RP should only raise `Program_Error`.

The session discussed the relationship between the RP and exceptions. Although the RP is silent on exceptions, it does currently require exceptions for the following error conditions: calling an entry when there is already a task queued attempting to suspend on a suspension object when there is already a task suspended issuing a potentially blocking operation from within a protected action violation of a ceiling priority trying to attach to a reserved interrupt

Consequently, Ravenscar requires exceptions; however if an application requires an exception-free environment then program verification techniques can be used to show that the Ravenscar runtime will not raise an exception. Some of these verification techniques might require program-wide flow control analysis.

It was noted, that the Ada Reference Manual indicated that calling a potentially blocking operation inside a protected action is a bounded error, and, consequently, raising program error was only one possible option available to the implementers. The Workshop unanimously supported the following recommendation:

Recommendation: Ravenscar requires potentially blocking operations in a protected action to be detected and Program Error to be raised. The only exception to this is if a subprogram call is to a foreign language domain (via `pragma Import`). In this case it may not be possible to detect the error.

3.2 Memory Management

Stephen Michell's position paper had raised the issue of whether the RP should remain silent on the use of dynamic memory management. The session discussed the issue from two perspectives: the implementation of the RP itself, and application programs using the RP. The following recommendation was supported unanimously by the workshop.

Recommendation The restriction `No_Implicit_Heap_Allocation` should be added to the Ravenscar lists of restrictions. Conformance to this restriction requires a compiler not to use the heap implicitly, and the run-time system not to use allocators.

One consideration in making this recommendation was that the ORK kernel currently used allocators. However, the ORK group believed that they could easily remove these from the implementation.

The workshop then reviewed its position on the use of allocators and storage pools at the application level. It considered three positions for the RP

- it should be silent on their use
 - it should specify no standard storage pools
 - it should allow optional use of standard storage pool
- After discussion, it was agreed that the RP should remain silent. However, there was no strong support for any of the above approaches. Note that the application is free to use program-defined storage pools.

3.3 Shared Variables

Stephen Michell's position paper raised the issue of whether the RP should say more on how `pragma atomic` should be implemented. It was agreed by the workshop that the RP should remain silent on this issue.

3.4 Interrupts

Currently the RP is silent on how interrupt handlers are attached. The session discussed the added complexity of allowing an application to attach interrupt handlers dynamically. The overriding issue was the ease of certifying such systems. The workshop believed that having dynamic handlers compromised certification and supported the following recommendation unanimously:

Recommendation: The RP should only support the static attachment of interrupt handlers via a new restriction "`No_Dynamic_Interrupt_Handlers`".

However, the type `Interrupt_Id` is defined in the same package as the subprograms that perform the dynamic attachment. Consequently, the restriction disallows a call to any subprogram declared in `Ada.Interrupts`.

3.5 Controlled Types

In Stephen Michell's position paper, the issue of whether controlled types should be prohibited by the RP was discussed. The motivation for disallowing them is to reduce the complexity of finalisation. An alternative view emerged that perhaps the restriction should be "no nested finalization". However, the RP says nothing about the sequential language and consequently allows an application (if it chooses) to use the OOP facility. For this reason, it was agreed by the workshop that the RP should remain silent on the issue.

3.6 Priority Range

Tullio Vardanega's position paper raised the issue of whether the RP should require more than the minimum range of priorities defined by the Ada Reference Manual. However, it was pointed out that the RP was about restrictions not extensions. Furthermore, allowing a larger range would potentially make RP non portable to non RP-compliant run-times and would not be implementable on top of POSIX.

3.7 Elaboration Control

Brian Dobbing raised the issue of how to control the interaction between task execution and package elaboration. He used the following (contrived) example to illustrate the issue.

```
package P is
  type Device_Type is ...;
  task Device_Handler is
    pragma Priority (High);
  end;
end P;

with Device_Config;
pragma Elaborate (Device_Config);
package body P is
  D : Device_Type;
  task body Device_Handler is
  begin
    loop
      Process (D.Current_Data);
      Next_Frame := Next_Frame +
                    Frame;
      delay until Next_Frame;
    end loop;
  end Device_Handler;
begin
  Device_Config.Initialize (D, ...);
end P;
```

The above example illustrates the basic problem that the Ada rules allow a task to begin to execute before the full

program has elaborated. This results in a race condition; hence, the program could access global data before it has been initialised.

The workshop had a heated discussion on several solutions to this problem, including: a general requirement for all global data to be initialized atomically at startup, the use of the priority model and programming paradigms to ensure no conflict.

Brian Dobbing proposed that it should be a bounded error for a task body or an interrupt handler to have an external effect, or to access an object that is external to its scope, before completion of all library unit elaboration. Possible effects of violation are:

- external effect occurs as defined by the Reference Manual
- external effect occurs after library unit elaboration is complete

One way of achieving the latter is to only activate tasks after the program has elaborated. If the program obeys the restriction, there is no detectable difference from the usual Ada activation rules. If it doesn't then it is a bounded error and the required effect is observed.

In general, there was limited support for this proposal. One view was that it was changing the Ada rules and, therefore, not supportable. Another view was that this was a general Ada problem and that it was not the job of the RP to solve it.

Associated with this issue is the problem of attaching interrupt handlers. The delivery of an interrupt ideally should not occur before the program has been elaborated.

It was agreed that Brian should propose an AI on the whole topic. The proposal should be for the revised ARM to allow an optional restriction to the RP to allow a program to activate all tasks at the beginning of the environmental task. This was unanimously supported by the workshop.

3.8 Floating Point

A discussion was held on whether the RP ought to support a pragma which allowed a task (or an interrupt handler) to specify that it made no use of floating point numbers. The motivation for this was to help the RP implementation optimize context switches. It was agreed that the compiler and run-time should be able to perform such an optimization without this help.

3.9 Pragma Ravenscar

The workshop gave unanimous support to a proposal that the RP should be in the new ARM standard, and overwhelming support (one abstention) to a proposal to have a pragma Ravenscar to match the profile.

4 Extending the Model

The workshop considered a number of possible extensions to the RP profile. The approach was not to make explicit extensions but rather to indicate that particular extensions would be useful and potentially worth the added complexity required for their implementation (indicated as POSSIBLE below).

It was agreed that this workshop would not consider new profiles but that this was a challenge for the next workshop, once position papers on other coherent profiles had been received.

4.1 Dynamic Priorities

The position paper by Audsley, Burns and Wellings had reported that dynamic priorities would increase the expressive power of the RP at little extra cost. The main motivation for this has come from trying to implement the ARINC APEX interface using the RP. Also, to implement EDF scheduling required dynamic priorities.

It was agreed that this required a RP-compliant kernel to be modified and that there would be a small distributed cost but that the implementation was simple. However, caution was expressed as such a facility made application programs more difficult to analyse and verify.

The workshop felt that such an addition to the RP was POSSIBLE.

4.2 Dynamic task creation

The position paper by Audsley and Wellings indicated that dynamic task creation would ease the implementation of a Distributed System Annex for the RP. Other motivations came from applications that needed mode changes but required tasks not be present if they were not active in a mode.

The workshop agreed that in its full generality this was not supportable. However, a restricted version where tasks were only created at library level (i.e. the master of the task was a library level package) was POSSIBLE.

4.3 Entry number and queues

The paper by Audsley and Wellings indicated that either having more entries in a protected object or allowing entry queues would ease the implementation of a Distributed System Annex.

It was agreed by the workshop that this was a major change to the RP kernel and the proposal was not supported.

4.4 Task entries

A proposal from the floor, suggested that task entries would increase the expressive power of the RP greatly. However, this was not supported because of the extra cost and because it changed the RP communication model.

4.5 Conditional and/or timed entry calls

A proposal from the floor for the addition of condition and timed entry calls (in any combination) was not supported by the workshop. Although conditional entry calls had minor impact of a RP kernel, the increase in expressive power wasn't great enough to justify its inclusion.

Timed entry calls did significantly increase the expressive power but the impact on the kernel was considered too great to justify its inclusion.

4.6 Dynamic Protected Objects

A proposal from the floor to allow dynamic library level protected objects was supported as a POSSIBLE addition.

5 Further Information

A complete description of the Ravenscar Profile can be found at www.cs.york.ac.uk/~burns/ravenscar.ps