Tasking and Object Orientation

Stephen Michell, Session Chair
Maurya Software
29 Maurya Court
Ottawa
Ontario, Canada K1G 5S3
steve@maurya.on.ca

Joyce L Tokar, PhD, Session Rapporteur
DDC-I, Inc.
400 N. 5th Street, Suite #1050
Phoenix, AZ 85004
USA
jlt@ddci.com

The aim of this session is to:

- Identify the required language changes and their impact for fully using object-oriented programming in multi-tasking, real-time systems.
- Issues to be addressed include:
  - Extensible Protected Types (EPTs)
  - Dynamic Ceiling Priorities
  - Extensible Tasks

One of the goals of this session is to develop language change proposals that are suitable to Real-Time applications.

The motivation for this session is based on several factors including:

- The tasking/task synchronization model has some anomalies that prevent the use of otherwise useful features, or may cause expensive workarounds and fragile code.
- Such language adjustments may benefit the language as a whole, but this workshop as a real time Ada workshop should ensure that such paradigms do not harm real time programming in Ada, and state how such paradigms should help.

The first discussion topic in this session was the use of Dynamic Ceiling Priorities. This discussion was based on the position paper by Jorge Real, Albert Llamosi and Alfons Crespo. Jorge argued that Dynamic Ceiling Priorities should be included in Ada. There was a long discussion on how changes to the ceiling priority occur; the effects of such changes; when the effects are realized with respect to protected actions and task synchronization points. A key topic in the discussion was the effects of a change in a protected object's ceiling priority in conjunction with the requeue operation.

Two models of behavior were considered for the interactions between the change in the ceiling priority of a protected object and the requeue operation:

- The ceiling priority of the protected object is only checked on initial calls to the protected object and not on requeues; or
- The ceiling priority of the protected object is checked when there is a requeue with abort operation.

Discussion continued as the details of the requeue operation were evaluated. The issue of when to check the ceiling priority was an issue when considering a requeue to a different protected object. When requeuing to a different protected object, should the ceiling priority be checked? If so, what ceiling priority should be checked?

For example, in one model, the ceiling priority would be modified immediately if the priority of the protected objected is being raised. On the other hand, if the ceiling priority is being lowered, then the effect of this operation would be deferred. This particular model would require a lot of overhead in implementation and it is not consistent with the model for dynamic task priorities. In addition, this model is difficult to use due to the unknown state of the protected object.

An alternative model would be to wait until the state of the protected object become quiescent. Then adjust the ceiling priority of the protected object. In this model, the state of the protected object and the effect of the ceiling priority change are deterministic. In addition, this model is consistent with dynamic task priorities.

As the discussion continued, there was consensus on the usefulness of this feature (18-0-7). However, there are a number of issues that need to be resolved before a change to the language could be recommended. The proposed solution should not require an additional lock. It should minimize the potential for priority inversion. The behavior of nested protected objects also needs to be studied and worked out. Finally, the rules should be consistent with the dynamic task priorities and should result in minimum impact on the existing semantics for requeue, tasking, etc.

Jorge Real will serve as the contact point for people with an interest in this topic.

The session moved on to the discussion of Extensible Protected Types (EPTs) based on the position paper from A.J. Wellings, B. Johnson, B. Sanden, J. Kienzle, T. Wolf, and S. Michell. The motivation for this work is to provide better integration between the concurrency and object oriented features of Ada.

As motivation, Thomas Wolf presented some the experiences that he has had in using Ada with CORBA. Presently, he is using a single protected object as the lock manager for a tagged tree. This paradigm is an example of abstraction inversion; ideally, an extensible protected object could be used to provide locking at the right level within the class hierarchy. This capability is necessary for implementing multi-tasking on top of a CORBA server.

The basis proposal is to extend protected objects in a manner similar to tagged types. A key component to EPTs is the extension of barrier conditions. As a protected object is extended, the barrier condition on an entry that overrides a parent entry must strengthen the parent’s entry barrier condition.

This led to a discussion of the relationship between the parent type and the child type. A child can call a parent entry as a "procedure" and therefore the call is not potentially suspending. The question in this case is what to do about the parent's entry condition. One position is to ignore the parent's condition. The other is to view it as an assertion. The view is that this call is part of the child's entry call and therefore, there is no need to reevaluate the parent's barrier. A requeue means that there can be no post-processing after a parent call.

The proposed model appears to co-exist with the rest of the core Ada language model. However, more investigation and definition is needed in the area of generics and mix-ins, and the model has not been integrated with the features in the Real-Time and Systems Programming Annexes nor with the Safety and Security Annex. Hence, EPTs and interrupt handlers needs to be studied further as does the entire priority model.

Outstanding issues include:

- Whether a call to the parent entry should check the barrier and raise Program_Error when there is a failure; presently, the EPT model does not check the parent barrier;
- Post processing of a requeue to a parent's entry call should or should not be done; presently, the EPT forbids post-processing;
- Should abstract entries have a guard, an optional guard, or should the guard be forbidden; presently, the EPT model forbids guards on abstract entries;
- Are the mix-in inheritance characteristics sufficient?

Similarly, the ramifications of the implementation of EPTs has not been considered. The execution model needs to be thought through and understood with respect to the impact on locking and dispatching.

The workshop agreed that this was a sensible way to define Extensible Protected Objects (18-0-7). The workshop also agreed that this proposal needs to be integrated with the Real-Time Annex (18-0-7). Finally, there was strong consensus that a reference implementation should be done to determine the impact of this model on the language and the execution model.

The final discussion topic in this session was based on a short presentation by Thomas Wolf on Extensible Task Types. Thomas has been working with multi-tasking systems and CORBA and has found a need for generic formal task types and an access task type.

The workshop evaluated Thomas's proposal and considered whether the language should be extended for generic tasks -- there was not a lot of support for this idea at the workshop (8-2-16).