Testing Ada Annex E - Distributed Systems

Richard D. Powers
Texas Instruments
McKinney, TX

Abstract
With the addition of special needs annexes to Ada 95, the traditional reliance on Ada Compiler Validation Capability style testing may not suffice. This paper explores some alternatives for testing a portion of the Distributed Systems Annex that deals with the method used to communicate between multi-processors in a real-time Ada system.

1 Introduction
Users of Ada compilers have traditionally relied upon the Ada Compiler Validation Capability (ACVC) test suite to give a reasonable level of confidence in the compilers. Furthermore, Ada compiler vendors have relied on the ACVC suite to aid in their internal testing efforts. With the addition of domain specific annexes in Ada 95, there are some requirements that do not lend themselves to ACVC style tests. One of these areas is the Partition Communication Subsystem (PCS) in Annex E[1].

2 Remote Procedure Call Interface
Section E.5 describes the partition communication subsystem that provides facilities for supporting communication between the active partitions of a multipartition program. Implementations are required to use the package described to implement remote subprogram calls. The design goal of this approach is to allow compiler/run-time system customers to build their own version of this package. The reference manual even states: "A body for the package System.RPC need not be supplied by the implementation." There are two different things that need to be insured with respect to section E.5:

1. The implementation does generate calls to the System.RPC interface defined, as described. Vendors must not be allowed to support the rest of the Annex using their own custom System.RPC package specification. This would conflict with an important goal of this annex.
2. Remote Procedure Calls (RPCs) work correctly on top of some implementation of System.RPC.

A solution to this problem is to require a vendor to supply some implementation of System.RPC to use for the testing for Annex E. Note that there is no requirement that this package be implemented on multiple processors. The vendor could support (for validation purposes) a Distributed Systems Annex on a uniprocessor. Ideally, a vendor should also demonstrate RPCs on a distributed system. The ACVC test suite should also supply a dummy version of System.RPC for use in a few specific tests.

3 Example solution
A simple approach for the ACVC supplied System.RPC package is to have it short circuit the interpartition RPCs by having each ACVC partition contain both the sender and receiver for given communication. For instance:

```
package Remote is
  pragma Remote_Call_Interface;
  type MSG_Type is ...;
  procedure Receive_Remote_Message
    (MSG : out MSG_Type);
  procedure Send_Remote_Message
    (MSG : in MSG_Type);
end Remote;
```

Then the test calls Send_Remote.Message and Receive_Remote.Message:

```
with Remote;
procedure Simple_Test is
  My_Output_MSG : Remote.MSG_Type;
  My_Input_MSG : Remote.MSG_Type;
begin
  -- Build a message up
  -- in My_Output_MSG
  Remote.Send_Remote_Message
    (My_Output_MSG);
  Remote.Receive_Remote_Message
    (My_Input_MSG);
  if My_Input_MSG /= My_Output_MSG then
    -- Report failure
    end if;
end Simple_Test;
```

The test then would include a body for System.RPC that takes any call to Send_Remote.Message and wraps the parameters back around for use as the out parameter for a later call to Receive_Remote.Message.

A series of tests of this type, each containing their own System RPC body, could be developed to test different variations of Do_RPC and Do_APC. The same approach could be used to write tests that included some of the remote bodies and checked the operation of RPC.Receiver.

A more robust set of tests could be developed that required multiple partitions, but still relied upon an ACVC supplied package body for System.RPC. This
could be achieved through the use of shared memory or other available resources. This type of test would be more complex to create and use because it would have to be modified for different implementations.

4 Conclusion

Over time vendors will produce implementations of System.RPC built on the primitives of commercially available network communication systems. For instance, a vendor might sell a System.RPC package that runs on top of a TCP/IP based system and works with any Ada compiler. This will encourage the development of tests to validate the System.RPC package and allow the vendors to validate using their choice of System.RPC, much like they validate using their choice of underlying Operating System.

References