ADA 9X REQUIREMENTS WORKSHOP, 22-26 MAY 1989, DESTIN FLORIDA

13-06
1989

Position Statement: **Reducing unpredictability in Ada executions**

Franco Mazzanti Istituto di Elaborazione dell'Informazione, via S.Maria 46, 56100 PISA, Italy

Abstract

One of the most drastic examples of unsafety of a program execution is surely that of a program crash (an unrecoverable run-time error causing the abort of the whole program).

This kind of failure is possible also in the case of Ada, since it is one of the possible effects of an erroneous execution.

Much could be done, just from now, for reducing the risk of a program crash at run-time: At the level of the Ada standard definition the existing uncertainties and ambiguities can be resolved providing a clearer picture of the "dangerous" aspects of the language; at the user lever, safe programming styles could be adopted guaranteeing the absence of program crashes in the set of the possible program behaviours; at the language implementation level, Ada executives and RTS could be designed to avoid the occurrence at run-time of erroneous executions, and to recover at run-time from them raising an exception before any unrecoverable damage is done. All these goals, however, could be reached with far less effort, is the safety issue is considered for its importance in the development of the revision requirements for Ada 9X. In particular, the class of errors allowing the execution to become unpredictable could be actually reduced if further checks were required by the standard, or if safer implementation policies were directly required by the standard. Let us consider only a few cases:

- The evaluation of the name of a deferred constant could just raise PROGRAM_ERROR if the full declaration of the constant has not yet been elaborated (in a way similar to the call of a subprogram whose body has not yet been elaborated) (AI_00155, instead, requires to consider the execution as erroneous).

- The index check, e.g. performed when accessing a component of an array object to verify that the index value belongs to the range defined by the bounds of the array object, might be required to be never optimizable, (e.g when the subtype of the expression used as index and the index subtype of the array definition are the same subtype). At least in this case the attempt to use a particularly inconsistent undefined value would be detected.

As an optimum case (analyzed in [1]), the possibility of a program crash could really be limited <u>only</u> to the occurrence of a run-time error when the corresponding run-time check has been suppressed (or, possibly, when machine code or pragma INTERFACE is used).

If the safety of Ada executions were not required the standard, but only guaranteed by particularly "safe" implementations, programs relying on this characteristics risk to become dangerously implementation dependent, and in the end, a not portable programming style would be encouraged by the "safe" implementation itself.

[1] S&M: "Erroneous Executions in Ada: Towards Run-Time detection", Deliverable 2 of Task 4.6 of EEC MAP project n. 755 "SFD-APSE".

* Current address: *Istituto di Elaborazione dell'Informazione, via S. Maria 46, 56100 Pisa, Italy*