The Ada 2005 Language

Alan Burns, Alejandro Alonso and Juan Antonio de la Puente

Introduction

The Ada language is still in use in many application domains, in particular the safety-critical areas such as avionics and railway signalling. The definition of the language itself has gone through a number of versions since its inception in the mid 1980s; the latest being Ada 2005. In earlier versions the support for real-time embedded systems was weak with the concurrency model having a number of limitations. The Ada 95 version was a considerable improvement and did include a well defined set of primitives for undertaking fixed priority (i.e. essentially static) scheduling for non-adaptive applications. The research community, including members of the ARTIST community, has been involved in defining new language features that could extend the applicability of Ada, especially to the adaptive (more dynamic) domain of applications. Many of these features have found themselves incorporated into Ada 2005 (again due to the efforts of ARTIST members).

Overview

Ada 2005 is defined by an international standard. It is a standard procedural language that does not give direct support to high level abstractions such as synchronous timing behaviour or periodic threads, but it does all these (and many other) to be programmed. These features include an expressive concurrency model, full support for fixed priority scheduling (including priority ceiling for shared objects, dynamic priorities and ceilings, and asynchronous transfer of control for handling timing faults), EDF scheduling, CPU time monitoring and control, timing events and the primitives from which execution-time servers can be constructed. Ada 2005 also has incorporated interfaces that make it easier to program reusable utilities.

Some of the most Ada 2005 innovations are those targeting the real-time community, providing capabilities that are neither addressed by other programming languages nor supported by most industrial execution platforms. The most important of these new services are: execution time clocks and timers, group execution time budgets, timing events, dynamic priorities for protected objects, immediate priority changes, and the new scheduling and task dispatching mechanisms.

As part of a joint effort with AdaCore, the GNAT run-time library has been adapted to run on top of MaRTE OS, which is a real-time operating system that can also be configured as a POSIX-thread library for GNU/Linux. The first objective of this implementation is to provide a reference platform for GNU/Linux, fully compliant with Ada 2005, available for industrial, research, and teaching environments. Additionally, this platform can be used as a test bed before cross-development, providing a more user-friendly environment for testing and debugging.

Conclusion and Future

The recent focus of the Ada 2005 work has been the development of a collection of real-time utilities. The research question being addressed is whether the potential for expressive power and reuse that Ada provides can be realised. The language itself continues to be used in application areas such as those found in the safety critical area. The standardisation process continues with possible languages changes still being debated. However, in the short term the new features of Ada 2005 need to be fully evaluated and incorporated into technology demonstrations and real systems.

Links and References.

J.P. Barnes, Programming in Ada 2005, Addison Wesley, ISBN-10: 0321340787, 2006.

A. Burns and A.J. Wellings, Concurrent and Real-Time Programming in Ada 2005, Cambridge University Press, ISBN-13: 9780521866972, 2007.

General Ada information: www.adaic.com

Implementations from AdaCore: www.adacore.com