# Ada Conference UK 2006[1]

***K. Fairlamb***
*AdaCore, 8 rue de Milan, Paris 75009, France*
*email: sales@adacore.com*

**Figure 1 John Rowlands, BAE Systems, presents at Ada Conference UK 2006**

*March 2006 saw the welcome return of an Ada event in the UK. The Ada Conference UK 2006, operated by the Centre for Software Reliability (CSR) in collaboration with the Safety-Critical Systems Club, took place this year on 28 March at the award winning Lowry Hotel in the heart of Manchester.*

The focus of the event was the recent language revision, called Ada 2005, and its continued suitability for building systems where reliability, efficiency, and safety are critical. The success of the 2006 event, attracting around 120 attendees (maybe the largest professional Ada event in the world), proved that Ada is more than ever at the front of software developers' minds.

As the CSR pointed out in its announcement of the event, Ada continues to prove itself as the answer for many of today's most complex programming challenges – especially in the areas of real time, embedded and safety-critical applications and in particular as the need for robust and reliable software systems increases.

[1] This paper previously appeared in Ada User Journal, Volume 26, Number 2, June 2006; reprinted with permission.

The event provided an excellent opportunity for members from all sectors of the Ada community, both in the UK and from abroad, to meet, share ideas, and reinforce links. Ada professionals from all four corners of the UK were present with a wide range of industries represented.

The event included plenary sessions by eminent Ada experts Robert Dewar and John Barnes, plus a series of technical talks by leading industrial experts, the abstracts of which are provided below and videos of which can be found on the AdaCore website at www.adacore.com.

In addition, a stream of well-attended vendor talks ran in parallel to the technical talks and a broad range of leading Ada toolset and service vendors displayed their technologies in the exhibition hall.

The next Ada UK conference is already in preparation, so look out for a forthcoming announcement regarding dates and venue!

## Conference papers at Ada UK

- **Welcome to Ada 2005**

  *John Barnes, author of 'Programming in Ada 2005'*

  Ada 2005 is the latest chapter in the Ada story. Ada 95 was a huge leap forward from Ada 83. However, experience has shown that Ada 95 has a number of roughish edges. Ada 2005 is not such a giant leap forward but aims rather to round off Ada 95 and so provide the community with a really smooth programming language suited for the demanding applications of the 21st century. John explained the specific goals of the development and introduced the key new features of Ada 2005 and thus set the scene for the rest of the day.

- **OOP & structure control in Ada 2005**

  *Pascal Leroy, IBM*

  Object-oriented techniques and structure control are important in very large systems in providing flexibility and extensibility. This talk gave an overview of the numerous enhancements that have been made in this area as part of the Ada 2005 Amendment. These enhancements include topics such as: Java-like interfaces, which allow proper multiple inheritance and integrate OOP with concurrent programming; the prefixed notation, used by many other languages, which simplifies usage of complex OO architectures; type extensions in nested scopes, which make it possible to declare controlled types at any level; object factories, which make it possible to dynamically create objects of any type in a class; explicit syntax for controlling overriding, which improves the safety of OO programs; the addition of limited and private with clauses, which support mutually dependent type structures crossing package boundaries and allow finer-grain visibility control; and finally improved aggregates and function returns which make limited types more flexible and easier to use.

- **Programming & certifying Ada software on an ARINC 653 platform**

  *George Romanski, Verocel Inc.*

  Ada applications running in a partitioned Integrated Modular Avionics environment such as ARINC 653 constrain the programmer, but also provide greater flexibility. The Ada Tasking model may be replaced by the Process, Semaphore, Blackboard, Event and other synchronization and control mechanisms. Exception management if present, must co-exist with a Health Monitoring system. Processor-time, memory and shared resources must be robustly partitioned. This is accomplished through a configuration control mechanism. While this restricts what a programmer can do within a partition, an application may be split across several partitions, and different variants of the applications may co-exist on the same IMA platform. Multiple schedules and mode switches will then select which sets of applications should run and how transitions occur.

  An IMA system needs to be configured very carefully. Platform providers, system integrators and application developers must set up a contracting model which specifies the responsibilities for and ownership of system parameters. In a safety critical system such contract models are subject to the same certification criteria as the application programs themselves. As systems evolve and applications change, the cost of system upgrade will remain high unless the components, Ada and programs in other languages, can be treated as applications in this modular system. This reduction of cost will only be accomplished if the impact of change can be isolated to the components that change.

- **Real time issues**

  *Alan Burns, University of York*

  Ada 2005 has introduced a number of new features that aid the programming and analysis of real-time systems. These features include: the inclusion of the Ravenscar profile for safety critical real-time systems, CPU monitoring and accounting, budgeting for the execution time of groups of tasks, timing event for efficient time driven computation, and new scheduling polices. The latter policies being non-

premeption, round robin, EDF (Earliest Deadline First) and combinations of these policies. This talk reviewed all of these features and included examples of use.

- **Building safety-critical/certified applications with Ada**

  *Rod White, MBDA*

  Developing safety-critical and certified applications presents different sets of problems in different domains. This talk considered those that relate to the missile products of MBDA, typically characterized by a small platform, demanding performance and a harsh environment. It considered issues such as the use of Ada, runtime systems, software re-use and the role of off-the-shelf elements. It also considered the challenges for the future – Ada has been the preferred language for a considerable period, but it is becoming necessary to address the need to incorporate elements in other languages e.g. C – this introduces a new set of issues and concerns.

- **Demonstrating Safety-Critical properties of an automatic train protection system**

  *Robin Messer, Westinghouse*

  This presentation described work done in collaboration with Aerosystems International and showed how safety critical properties of an ATP have been:

  - Captured from hazard analysis
  - Analysed using a UML model
  - Translated in to SPARK annotations
  - Metrics captured on the work

- **Safety-Critical Software: Looking for an argument**

  *Carl Sandom, iSys Integrity*

  This presentation provided software developers with a broad overview of what an Independent Safety Auditor (ISA), safety regulator or third-party might look for when evaluating safety-critical software. The presentation should be of interest to anyone undertaking either safety-critical software development from the beginning or the retrospective safety assessment of software which has not been developed explicitly for safety-critical use but is subsequently used within safety-critical systems.

  Software safety assurance can be provided to a third party by constructing a clear and compelling safety argument which is underpinned with evidence from various diverse sources. The structure of the safety argument will determine the type and depth of the evidence that must be generated during development and/or collected in-service to support any claims made regarding the safety of the software in the context of its actual or assumed use.

  The provision of safety assurance was the central topic of this presentation and a pragmatic approach to the construction of a clear and compelling software safety argument was described in detail. The presentation was based upon a software safety assurance strategy that has been used to support system safety certification or acceptance for various real-life software development projects which the presenter has been directly involved with either as the ISA or as part of the safety assurance team.

- **Executable Modelling with UML and Ada: The X Factor**

  *John Rowlands, BAE Systems*

  Traditionally, executability is a property possessed by programming languages, but often not by design languages. For instance a simple UML design only captures the structure of a software system and provides a high level description of behaviour, enabling ease of navigation for maintenance. However, in order to improve the productivity of the software process, a rich model is needed that allows animation and code generation. Animation allows the design to be tested prior to committing to code or deploying to a particular platform. Full code generation allows the software to be maintained at the design level, lifting the level of abstraction at which the developer interacts with the design. However, if we are to maintain our software at the model level, we need to have access to all the features we have come to take for granted with traditional programming languages, such as ease of static checking, debugging and testing.

  In order to enrich a UML model for executability and code generation, an action language is needed. This language needs to understand the architectural concepts inherent in UML and add a detailed definition of the behaviour of the software. In the Ada community we are used to the idea that the programming language inherently provides support for finding errors early, such as strong typing, declaration before use and ease of static analysis. The ideal action language should allow the software engineer to work at the UML level of abstraction whilst providing similar static checking facilities.

The presentation addressed the question of how such an action language could be constructed, the features that it should exhibit and the way in which it could be defined.

- **Mixed Criticality**

  *Peter Amey, Praxis High Integrity Systems*

  High integrity applications, such as those performing safety or security critical functions, are usually built to conform to standards such RTCA DO-178B or UK Def Stan 00-55. Typically such standards define ascending levels of criticality each of which requires a different and increasingly onerous level of verification. It is very common to find that real systems contain code of multiple criticality levels. For example, a critical control system may generate a non-critical usage log. Unless segregation can be demonstrated to a very high degree of confidence, there is usually no alternative to verifying all the software components to the standard required by the most critical element, leading to an increase in overall cost. The presentation described the novel use of static analysis to provide a robust segregation of differing criticality levels, thus allowing appropriate verification techniques to be applied at the subprogram level. We call this fine-grained matching of verification level to subprogram criticality smart certification.

- **Ada 2005 & high integrity systems**

  *Robert Dewar, AdaCore*

  Ada has been, and continues to be, successful for Safety-Critical applications. This talk covered the foundations of the Ada language and its evolution being based on good programming practice and smooth integration of new features rather than specific technical capabilities. Among these readability, the package structure, the strong typing system, compile time checking, and run time exceptions all help to ensure that Ada continues to be used widely in Safety-Critical applications. The presentation concluded by emphasizing the importance of the Ada “culture” instilled in programmers.