# Session Summary: Safety Improvements for Consideration

Chair: Juan Antonio de la Puente

Rapporteur: Luís Miguel Pinho

## 1. Introduction

The objective of this session was to consider current work in the software safety area, and to discuss the possible impacts in the language. The chairman, Juan Antonio de la Puente, suggested that the session consider two different topics:

- Safety-related issues, and impacts/changes in the language
- A proposal for an Application Open Interface in the ARINC 653 standard

## 2. Safety-related issues

The first part of the session included three presentations, covering different aspects of the software safety area:

- Experiences with certifying VxWorks
- Software related accidents
- Using partitions for space-based security

### 2.1. Experiences with certifying VxWorks

In the first presentation, George Romanski described the process of certifying the VxWorks Real-Time Operating System (RTOS), and the associated Board Support Package (BSP), to attain DO-178B Level A certification. The goal was to be able to use the RTOS in a FAA ground-based application.

The main difficulty with this process was that the RTOS was an already developed product, being necessary to re-engineer the requirements and design phases from the existent code and documentation. In order to introduce a measure of the complexity of the process, George presented some numbers: the VxWorks RTOS is constituted by 122 files (approximately 12000 KLOC), while the BSP totalled 41 files (7000 KLOC). The re-engineering process resulted in approximately 1300 requirements for the RTOS and 700 requirements for the BSP.

The main part of the presentation was the description of the tools that were developed to assist in the certification process. The use of these tools allowed the actual auditing to be performed in 2 days (compared to 2 weeks of auditing of Aonix's C-SMART certification materials when they were paper based). Due to the FAA requirements, those tools had to be qualified.

Particularly important, was the developed requirements database and associated tool (VeroTrace), which allowed to maintain the traceability of all the certification process, and to generate XML tagged files for all the documents necessary for the certification process. One interesting point was that it was necessary to maintain a record of all review comments, even those that had already been addressed (some of them were side comments in the paper documents).

Another tool, VeroCode, performed the verification of the object code, allowing to decrease the effort of coverage analysis (if only short circuit operations are used as in VxWorks). The experience was that, by using short circuit operations, Modified Condition/Decision Coverage (MCDC) was easier. However, some care had to be taken as compiler optimisations changed the generated code.

Code coupling was also difficult since the linking image changed each time a new test application was linked. This was dealt with the VeroLink tool, that checked the link map and all links resolved by the linker (there were some doubts about the process, which were resolved by a FAA issue paper).

One of the difficulties found during testing was due to the bus snooping feature of the Power PC architecture that is used instead of cache-safe buffers (to guarantee data coherency when devices access it). This feature was not expected and it was necessary to review the requirements and the code, and to develop new test cases.

As a final comment, George stated that this certification process was more complex than the

certification of Aonix's Raven, due to the higher complexity induced by the C language and greater functionality of the OS.

## 2.2. Software Related Accidents

In the second presentation, Kristina Lundqvist presented some results of the work being carried out concerning the modelling of software accidents. The presentation focused on the main types and common factors of software accidents.

After an introduction describing some accident real-cases (particularly the JAS 39 plane accidents), Kristina stated that accidents have changed their nature. The principal cause is no longer software "bugs", but flawed requirements. Particularly, incomplete or wrong assumptions about the expected behaviour of the software, system states that were not handled or unexpected environmental conditions.

The work considered accidents to be of two main types:

- Component failure accidents;
- System accidents (arising from interaction between components – not from a component itself).

Kristina also noted that this last type of accidents is caused by the complexity and tight coupling of components and is worsen by the increased use of computers.

The work also allowed determining that there are some common factors in the spacecraft accidents:

- Flaws in safety culture, such as overconfidence, over relying in redundancy, underestimating software risks.
- Ineffective organisational structure and communication, related to, for instance, diffusion of responsibility or authority, limited communication channels.
- Ineffective/inadequate technical activities, such as inadequate specifications, operational person, not understanding the automation, flawed review process, test and simulation environment that do not match the operational environment.

As a conclusion, Kristina stated that this change of causes in software-related accidents means that safety must be built-in in the process of development and the current standard processes must be modified.

## 2.3. Using partitions for space-based security

The final presentation of the first part of the session was made by Bruce Lewis, concerning the use of space and time partitioning (STP) in avionics systems. The importance of discussing STP comes from the fact that the FAA will be requiring civil compliance (DO178B) from military aircraft that use civil air space or resources. Europeans may have a similar rule. Hence a number of military aircraft programs are working toward certification. This is bringing a significant interest in space and time partitioning as an approach, currently used by commercial aircraft, to reduce system verification costs. STP protection enables applications with different safety levels to be deployed in the same processor (reducing HW costs and weight) without requiring verification to the highest safety level of any of the processes on the processor.

Bruce started by presenting the current work with MetaH, which is undergoing a standardisation process to become an Avionics Architecture Description Language (AADL). The MetaH language (and toolset) is targeted to the development of reliable, real-time multiprocessor avionics system architectures. It allows to capture and specify the different views of the system being developed (safety levels, security levels, modes of operation, etc.).

Bruce then presented the results of several ports of a generic missile architecture, which had been done using MetaH. The experiences obtained allowed to conclude that the combination of Ada 95 and MetaH significantly improved the upgrade/porting costs comparing with the current approach (error-prone and paper dependent). The previous Ada 83 implementation, although producing an efficient implementation, did not provide an equally portable solution.

Another conclusion that could be obtained was that STP is an important issue in order to meet the new avionics requirements. Bruce's view was that Ada should support time partitioning via adding a CPU timing capability and that Ada should also support space partitioning via separate address spaces similar to the Distributed Systems Annex (DSA) through a simplified real-time DSA. In his point of view, this is an opportunity for the language in the real-time safety critical market. Otherwise, applications will have to use other support (for instance, POSIX), that may not provide the same efficiency as using Ravenscar and a stronger partitioning model.

He then introduced the two time partitioning models that he would like the workshop to consider:

- Static non-preemptive partitioning: partitions are allocated a fixed amount of the computing resources, and the process to execute is the one with the highest priority in the currently active partition;
- Preemptive fixed priority partitioning: processes are scheduled globally between partitions, and the process to execute is the one with the highest priority in all partitions.

He considered that the latter allows a more flexible approach to the development of partitioned systems.

Answering to George that noted that the industry is using the former, Bruce counter-argued that currently there is dissatisfaction in the industry with the scheduling complexity of the former, that Honeywell's business jet line in now using the latter and he predicted future movement toward the simpler approach. CPU timing, the approach used to accomplish time partitioning in the latter, is also of significant value for advanced real-time scheduling and has recently been added to POSIX for that reason.

From the discussion that followed the presentation, several issues were raised (there were no conclusions):

- It is important for Ada to address this issue, since this is an important market for Ada;
- It is not clear if this should be considered within the current Distributed Systems Annex (DSA), or to think of a new DSA;
- Maybe both models of time partitioning could be supported.

## 3. ARINC 653 Revision

The second part of the session was devoted to the discussion of a proposal George Romanski is making within the ARINC 653 review process. This process is not only to correct "bugs" in the specification, but also to consider new features.

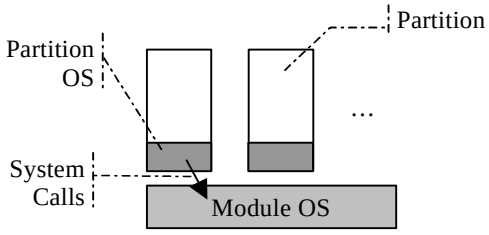


**Fig. 1 – ARINC 653 Model**

The ARINC 653 defines a general-purpose interface (APEX) between the operating system of an avionics computer resource and the application software. It specifies the support to the execution of different partitions within the same processor. Each partition is supported by a partition operating system, and a processor module operating system is responsible for interfacing with the hardware and for the management of the set of partitions (Figure 1).

The problem is that all existent applications must be re-written using the APEX interface, which makes their migration to a different architecture more difficult.

George's proposal is to create a minimum standard interface (the Application Open Interface) between applications and the module OS, which would only specify a BSP-like API, mapping the common interface currently used for applications accessing the hardware: entry points, memory requests, clock interrupt, time, I/O and error data. This would allow current code to be transferred to APEX without needing re-write, also facilitating the co-existence of different languages and scheduling models.

After the presentation several issues were brought into the discussion, although the group did not reach any conclusion:

- The idea in the proposal is to specify only the inter-schedule of partitions, while within a partition scheduling would be application-dependent.
- The idea in the proposal is interesting. The current approach has failed in APEX, and a different approach is needed;
- To support partition replication, the current defined interfaces of the module OS must be kept. Particularly, since the module OS will probable need to call the partition OS;
- There are some concerns (applied to both the current and the proposed approaches) of the safety of data going to a non-active partition. Since it is kept on the module OS, problems can arise if there are partitions with different criticality levels.