

GUIDELINES FOR MODELLING THE ARINC 653 SOFTWARE ARCHITECTURE

Roberto López, Natalia Casas, Jesús Fornis
GMV AEROSPACE AND DEFENCE
Isaac Newton, 11 P.T.M. Tres Cantos
28760 Madrid, Spain
Tel. +34 91 807 21 00
Fax. +34 91 807 21 99
{roblopez, nacasas, jfornis}@gmv.com

Ismael Lafoz, Miguel Ángel Mozas
EADS-CASA
Paseo John Lennon s/n
28906 Getafe, Madrid, Spain
Tel. +34 91 624 18 79
Fax. +34 91 624 27 05
{Ismael.Lafoz,Miguel.Mozas}@casa.eads.net

ABSTRACT

The nature of an Arinc 653 system makes the design of the software different from a classical system. From the architecture point of view, the main difference is the appearance of the partition concept, which involves new design criterions. The advantages of a partitioned system are many, but the increase of integration costs and the loss of performance should be considered.

In order to accurately design the partitions of the system, our main objective has been getting the optimum balance between the advantages of a partitioned system (reduction of certification effort, improvement in safety, reusability, scalability and testability) and the related development and integrations costs.

Another challenge is how to model and represent the Arinc 653 architecture because UML does not provide any means to reflect the new design based on partitions, ports, etc. Using a modelling tool greatly improves the whole development process.

Based on our experience in the development of Arinc 653 critical systems, this paper presents the guidelines to design the partitions of an Arinc 653 architecture, a Partition-Based-Design, and proposes a solution to represent this architecture by using a new UML profile, UML-653 profile, to represent all the Arinc 653 concepts.

In analyzing our partition-based software system it must be taken into account both the available resources, such as time budget, CPU and memory, and the system requirements, such as Function-DAL, and the flexibility and fault tolerance in them.

The goal of the design of Arinc 653 software architecture should be in our opinion to reach the best balance between the use of the available resources and the compliance of the system requirements, by considering the positive impact on quality factors from the use of partitions.

There are five main quality factors, which would benefit from the use of partitions:

- Reduction of Certification Effort: The isolation between SW with different DAL assignments allows the application of different processes according to the needs of each SW partition.
- Increase in Safety: The isolation between partitions prevents failure propagation.
- Increase in Scalability: The isolation between partitions reduces the needs of regression testing in case of a SW change and allows the introduction or removal of SW without changing execution conditions.
- Increase in Reusability: Reinforces well-defined SW component functionality (suitable for reuse), which improves SW reusability.
- Increase in Testability: The clearly defined and controlled interface of a partition, besides the standard execution environment definition, makes the testing process easier.

However there are negative impacts such as:

- Increase in design cost: The higher design cost must be considered in terms of data coupling, synchronization, coherence and complexity of interfaces between partitions.
- Loss of Performance: Increasing the number of partitions produces a loss of performance due to various issues: the time and the memory assigned to a partition must support the worst scenario of execution, producing an inefficient use of the resources when increasing the number of partitions. Besides, additional time is needed due to switching partition time and inter-partition communication.

To find the best IMA Architecture design, we propose the iterative procedure to face a trade-off between the quality factors and the negative impacts. The proposed iterative procedure is shown in Figure 1. At each decision point (related to the explained

quality factors), we must bear in mind the considered parameters, the system requirements and the available resources.

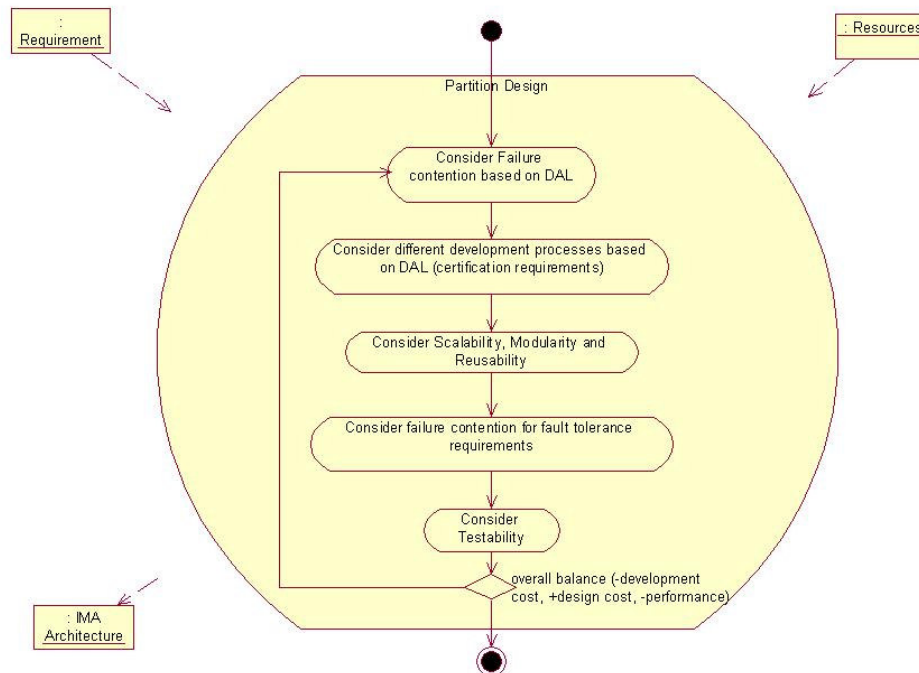


Figure 1: Partition-based Arinc system – Trade-off between quality factors and requirements

In order to represent a Partition-Based-Design, a new UML profile is proposed. This profile has been developed due to the difficulties of reflecting in the model all the Arinc-653 architecture composed by partitions, processes, communication between partitions, etc. in an integrated way with the rest of the UML design. Also, since the nature of the system life cycle is close to a Prototyping Model, it is necessary to develop a tool to quickly re-configure the architecture of the system when required. In order to achieve this objective, an automatic code generator was developed to build the complete configuration of the system.

The deployment of the architecture is represented through different class diagrams, where all the Arinc-653 concepts are presented as stereotyped classes (module, partition, sampling port, queuing port, module schedule, partition schedule, window schedule, partition health monitor table, etc). The semantics of the stereotypes and relationships are the same as stated in the Arinc-653 standard specification. Each one of these extended classes have specific attributes or properties in order to configure the whole system as described in the Arinc specification.

The architecture view is integrated with the classical logical design based on UML classes by means of two special relationships: a dependency between an Arinc-653 process and a main process class, and a link between the Arinc-653 communication classes and a wrapper interface that represents these communication in the logical design. The main process class from the logical design must provide two methods, a “preamble” one for initialization purposes, and a “one step” procedure which is called in each cycle of a periodic process, or only once if the process is not periodic. On the other hand, the wrapper interfaces hide the Arinc-653 communication details,

providing a well-defined layer that isolate the application code from the low level details. By using this layer, the different processes do not have to worry about the communication mechanism, or if this communication is intra or inter partition. The layer provides an easy way of reconfiguring the architecture, as the impact in the application code is minimized.

The automatic code generator tool minimizes the effects of change impact, providing a powerful mechanism for directly generating from the UML model the following source code:

- The interface wrappers described above, that hide the communication mechanism.
- The partition entry points, that creates all the processes, communication ports, etc. with the information extracted from the corresponding diagrams.
- The classes for representing Arinc653 processes.
- The XML configuration of the whole system that will be used to build the complete system image.

Besides this advantage, there are lots of benefits as a consequence of modelling the ARINC-653 architecture:

- The possibility of having a graphical view of the architecture is an easy way to understand and transmit the system deployment.
- A better description of the system configuration, instead of using the corresponding XML tables where the implementation details are shown at the same level as the abstract aspects.
- As a consequence of the integration of the architecture with the rest of the design, it is possible to make a coherence check of the whole system at design level, which reduces possible development errors.
- The automatic generation of the configuration and the source code.