Preliminary Design of Future Reconfigurable IMA Platforms

Pierre Bieber
ONERA, Toulouse France

Eric Noulard
ONERA, Toulouse France

Claire Pagetti
ONERA, Toulouse France

Thierry Planche
Airbus, Toulouse, France

François Vialard
Airbus, Toulouse, France

ABSTRACT

The next generation of IMA platforms should include reconfiguration capabilities in order to limit the effect of hardware failures on aircraft operational reliability. In this paper, we investigate architecture principles for such platforms and propose adequate reconfiguration services. A preliminary analysis of the design feasibility and its contribution to operational reliability is performed. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 / 2007-2013) under Grant Agreement n°ACP7-GA-2008-211439.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Design studies, Fault tolerance, Measurement techniques, Modeling techniques

General Terms

Design

Keywords

IMA, reconfiguration, operational reliability.

1. INTRODUCTION

The trend for modern military and civilian aircrafts is to support aircraft applications with an Integrated Modular Avionics (IMA) platform. The current generation of IMA (IMA-1G) replaces separate and dissimilar equipments, with fewer common processing modules, sharing the necessary communication links. The number of processing units in the A380 is half that of previous generations. Reductions in airline operating costs are expected to be significant. Indeed, the decrease of the number of computers and cables (for power supply or communication) contributes to the reduction of the aircraft weight that leads to a better fuel consumption efficiency. The reduction of the number of types of equipments will help the airline to buy and store less types of spare parts, this should lead to maintenance savings.

A typical IMA platform is described in Fig. 1. Its hardware architecture is made of a set of computing modules (numbered 1 to 5) that are connected to communication switches (labelled S1 to S4). Computing modules are grouped into clusters, such that all computing modules in a cluster are connected to the same communication switch.

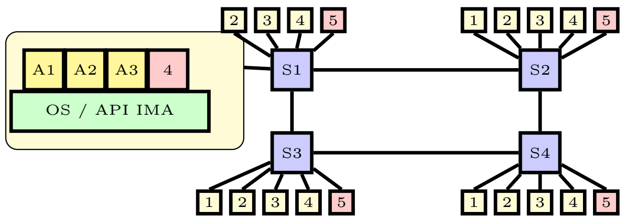


Figure 1: Architecture of Reconfigurable IMA Platform

Avionics applications (labelled A1 to A3) are hosted in the partitions running on the computing modules. Data flows exchanged by applications hosted on different computing modules are transmitted through communication switch paths that connect the two computing modules.

1.1 General objectives

The SCARLETT project[1] funded by the European Union and led by Thales as a coordinator is currently studying the next generation IMA. Among other objectives, the project aims at adding reconfiguration capabilities to the IMA-1G platform. Fig. 1 shows in red the new spare computing modules and partitions that are added to the platform architecture. Reconfigurable IMA should be able to change the configuration of the platform by moving applications hosted on a faulty computing module to spare computing elements. The main goals to be achieved by the reconfigurable IMA platform are:

1. *Improve the operational reliability of the aircraft while preserving current levels of Safety.*
2. *Avoid unscheduled maintenance and associated costs.*
3. *Limit the impact of reconfiguration on certification practices and effort.*

[1]http://www.scarlettproject.eu/, Point of contact: didier.hainaut@fr.thalesgroup.com

Aircraft systems have to enforce stringent safety requirements that address the effects of failures on the life of passengers. To fulfill these requirements a minimum level of redundancy is associated with an application on the basis of the severity of the effects of its loss. For instance, three occurrences of an application managing cabin air pressure would be required because loss of cabin pressurization is catastrophic whereas no occurrence of an application managing in-flight entertainment is required as it is considered as a comfort application whose loss has no safety effects.

Operational reliability addresses the effect of failures on economical aspects of flight operations. One source of improvement is to decrease the number of flight delays or cancellations caused by faulty computing modules. Before each flight, the health status of all equipments is assessed in order to check whether for all applications the correct level of redundancy is available. If this is not the case the aircraft cannot be used (it is NOGO). It should be possible to restore the minimum level of redundancy by moving the applications running on the faulty module to a non-faulty one. This should also help to defer maintenance operations until the aircraft has reached an appropriate location.

Aircraft manufacturers have to show compliance with international regulations using means that have been accepted by the certification authorities. This includes showing that safety requirements are enforced, establishing the predictability of communication and computing real-time performances and developing software and hardware according to strict development guidelines.

1.2 Reconfiguration Scenarios

A number of reconfiguration scenarios have been considered and assessed by the SCARLETT consortium. We use three attributes to present the scenarios :

- **Granularity level**: reconfiguration is either performed at module or at partition level. For module level reconfiguration, a spare computing module is allocated to all applications running on the faulty computing module. For partition level reconfiguration, spare partitions running on non-faulty modules are allocated to the applications running on the faulty computing module.
- **Location**: reconfiguration can be performed either locally on modules belonging to the same cluster than the faulty module or distantly on any module of the platform.
- **Time**: reconfiguration can be performed during the flight or on ground when the aircraft is stopped.

An important benefit of local reconfiguration over distant reconfiguration is that it would require almost no reconfiguration of communication equipments whereas distant reconfiguration would involve reconfiguration of the communication equipment. This would require major evolution of the current technology used to support communications.

The loss of applications when the aircraft is stopped has generally very little effect on safety. So it is likely to be much simpler to show the innocuity of reconfiguration on safety when it is performed on ground rather than during flight.

1.3 Outline of the paper

In the following of this paper, we first describe the services that a platform should offer in order to be able to reconfigure itself. The services are defined through a number of basic steps. Then, a preliminary analysis of the proposed design is presented.

2. RECONFIGURATION SERVICES

Several functions are available in order to manage the platform, they include:

- Data-Loading function that stores all the application software and loads the application software according to the allocation of applications onto the computing modules.
- Monitoring and Fault Detection function that constantly receives information about the health of the hardware components and is able to detect that a component is faulty.
- Power Supply Management function that is able to switch on and off the power supply of the various hardware components of the platform.

For reconfigurability purpose, a new function, called the *Reconfiguration Supervisor* (supervisor for short), is embedded in the aircraft. The role of the supervisor consists in determining when a reconfiguration can occur and in performing a “correct by construction” modification of configurations in order to reach a better and safe state. The supervisor behaviour is described below.

2.1 Triggering a reconfiguration

When a computing module fails, a reconfiguration can be launched if this failure has an operational reliability impact, meaning that the aircraft becomes NOGO. The *Monitoring and Fault detection* function detects a NOGO module failure and sends this event to the supervisor. First, the supervisor applies usual maintenance procedure to check that the failure cannot be repaired by a simple reset of the module. For this, it interacts with the *Power Supply Management* and the *Monitoring and Fault detection* to check if the reset has repaired the module. If the reset works, the module restarts the hosted avionics applications and the failure is corrected. Otherwise, the supervisor performs the next steps.

2.2 Selection of a correct configuration

When the failure is confirmed, the supervisor must determine the current state of the platform in order to select the next configuration. The sorting takes into account the side-effect on aircraft and the duration of the reconfiguration execution. In the following, we describe the selection process.

2.2.1 *Set of configurations*

We note *Application* the set of applications hosted by the platform, *Cluster* the set of clusters of the platform, *Basic_Module* (resp. *Spare_Module*) the set of modules (resp. spare modules) in a cluster and *Basic_Partition* (resp. *Spare_Partition*) the set of partitions (resp. spare partition) running on a module. Let us denote $bm = |Basic\_Module|$, $sm = |Spare\_Module|$, $c = |Cluster|$, $f = |Application|$, $bp = |Basic\_Partition|$ and $sp = |Spare\_Partition|$.

DEFINITION 1 (CONFIGURATION). *A configuration is an allocation of avionics applications on computing elements, it is represented by a function*

$$Conf\colon Application \rightarrow Computing\_Element$$

where the set Computing_Element is defined by $Cluster \times (Basic\_Module \cup Spare\_Module) \times (Basic\_Partition \cup Spare\_Partition)$.

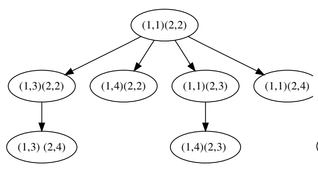
For module level reconfiguration, the identifier of the partition for an application remains unchanged wherever the application is allocated. Therefore, we can optimise the set to be $Computing\_Element = Basic\_Module \cup Spare\_Module$.

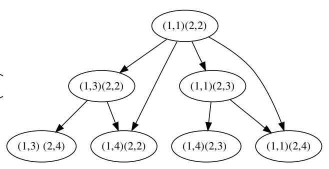
In Fig. 1, avionics application A_1 is allocated in its initial partition on the first module of the first cluster. Thus $Conf(A1) = (1, 1, 1)$. For module level reconfiguration, there are exactly $(c \times (bm + sm))^f$ configurations. This corresponds to the number of integer functions $[1, f] \rightarrow [1, c \times (bm + sm)]$. For partition level reconfiguration, there are exactly $A^f_{c\times(bm+sm)\times(bp+sp)} = \frac{(c\times(bm+sm)\times(bp+sp))!}{(c\times(bm+sm)\times(bp+sp)-f)!}$ configurations. This corresponds to the number of injective integer functions $[1, f] \rightarrow [1, c \times (bm + sm) \times (bp + sp)]$. All these configurations are known at design time. Note that we only consider nominal configurations and not degraded ones where an avionics application may not be allocated because there are not enough fail-free computing elements. The sequences of possible reconfigurations starting from an initial configuration can then be represented by a directed acyclic graph.

2.2.2 *Reconfiguration Policy*

The reconfiguration policy defines generic rules to be followed. It is chosen off-line and impacts the selection process of the next configuration. For instance, we can decide that there is no priority among avionics applications and then, once a spare has been occupied, no other application can be hosted on this spare. On the opposite, we could associate a priority level with the applications. Then, a reconfigured application can be removed to leave the spare to a failed application with higher priority level. Another rule can give an order on the spares. The policy would then consist in reallocating on the first spare if it is available, otherwise on the second if this one is available and so on.

Let us consider the architecture f=2, c=1, bm=2, sm=2 where a module can host at most one application. For a module level reconfiguration, we obtain several directed acyclic graphs of reconfigurations depending of the policy. For any of these graphs, a node corresponds to a configuration which is a list of pairs (number of application, number of hosting module). We have drawn below the graphs associated with two policies that both order spare modules. On the left side, spares are not reconfigured (meaning that if a spare fails then hosted applications are lost) while on the right side they can be reallocated. Reconfiguration graphs are different. For instance, the transition from $(1, 3)(2, 2)$ to $(1, 4)(2, 2)$ is missing on the left side graph because when application 1 is running on spare 3 and this spare fails it is not possible to move 1 to spare 4.





The policy acts on acceptable reallocations: it determines the number of reachable states and the number of transitions. The selection of the next configuration consists in choosing a safe reachable state and safe transition. Note that there may be no acceptable target reconfiguration.

2.2.3 *Resource and Real-time constraints*

A configuration is safe if it satisfies some constraints. For instance, an application can be hosted on a module only if it provides adequate resources for the application such as processing power or memory. There are other kinds of constraints, such as segregation, that are described in [6].

A transition is safe if the intermediate steps are safe (they do not impact the integrity of the aircraft) and the duration of the transition is bounded. The transition from one configuration to another is done by applying several basic procedures. All such elementary procedures are stored in some repository attached with a WCET (worst case execution time). For instance, data-loading a complete module is an elementary step and always takes less than a bounded amount of time which is computed off-line. Globally, the reconfiguration execution on ground must take less than 15 minutes to limit the flight delay.

2.2.4 *Continuity of service*

When a module fails, the other avionics applications should not be impacted by a reconfiguration. In case of partition level reconfiguration, the interactions between the supervisor and modules involved in the current reconfiguration must be transparent for the other partitions. For instance, data-loading and initialisation must be realised during the partition time. In case of distant reconfiguration, the routing of the impacted switches must be modified while ensuring the continuity of the remaining traffic.

2.3 Reconfiguration execution

If a correct transition, with respect to the avionics constraints and the reconfiguration policy, has been found, the reconfiguration is performed. Basically, a reconfiguration is decomposed in elementary steps: power up of a spare module, test that the spare is fail-free and load of the time application sequencing (at module level), *data-loading* of the code and initialisation of the partitions on the spare (during predetermined time slots for partition level), verification by the *Monitoring and Fault detection* that the spares are correctly working. A notification and report are provided to the maintenance terminals.

The sequence execution should be transactional and secured, i.e. the sequence should entirely succeed or totally fail. For this purpose, each step is acknowledged (succeeded, failed, not performed). Any step can be aborted without side-effect on aircraft safety, performance and security. For instance, an access to an already allocated memory must be refused by the *Monitoring and Fault detection*.

3. PROPOSED SOLUTIONS

3.1 Architecture Platform

A module is composed of a generic calculator and a generic operating system compliant with Arinc 653 [2]. The Arinc API provides spatial and temporal segregation for avionics functions. Spatial segregation is enforced thanks to an off-line segregated memory address allocation. Temporal segregation is enforced by a predetermined temporal scheduling of avionics applications. In its allocated time slot, an avionics application accesses to the CPU and computes its data. At the end of the slot, the operating system preempts the application, empties the shared resources (such as the cache for instance) and awakes the next avionics application.

Centralised Solution The reconfiguration supervisor function is allocated on one module and pilots all the reconfigurations. This solution will be developed on a real demonstrator in the Scarlett project for module level reconfiguration and a simple policy.

Distributed Solution The reconfiguration supervisor function is distributed on a module in each cluster. Each local supervisor can independently realise a local reconfiguration. A master has in charge to keep a global view of the platform configuration.

Whatever is the solution, to enable reconfiguration, there must be some improvements which SCARLETT expects to make available through platform services. Among them, it must be possible to load some code and to make some initialisation without a maintenance intervention and while other avionics applications are running.

3.2 Graph of Reconfigurations

In the following we restrict ourselves to module level reconfiguration using the two policies described in the previous section. They both order the modules and the first one disallows reconfiguration of spares while the second reconfigures the spares. For the demonstrator, we have implemented a software that computes the directed reconfiguration graph for a given architecture and a given policy.

3.2.1 *Policy without Spare Reconfiguration*

The simple policy produces a tree. The number of nodes and the number of transitions from an initial configuration can be computed:

	number of configurations	number of transitions
local	$NB_l = \left(\sum_{0\leq k\leq sm} C^k_{bm} A^k_{sm}\right)^c$	$NB_l - 1$
distant	$NB_d = \sum_{0\leq k\leq (c\times sm)} C^k_{c\times bm} A^k_{c\times sm}$	$NB_d - 1$

PROOF. Note that the ordering of occurrence of failures leads to a specific configuration and that a spare may fail. There is an initial configuration. When one single failure occurs, the avionics functions hosted by the module can be allocated on the first spare module if it is fail-free, otherwise they are allocated on the second spare and so on. Thus a module can be re-allocated on the p spares and all the modules can fail. This entails that there are $n \times p$ reachable configurations from a single failure. For a k failures situation, there may be A^k_p possible reallocations depending on the spares state and the ordering of failures, whereas there are C^k_n such situations.

We now compute the number of transitions. We prove that a node (except the root) has a unique parent. Let us consider a node with l occupied spares. For instance the failed modules are $i_1, \ldots, i_l$ which are currently re-hosted $(i_1, s_{j_1}), \ldots (i_l, s_{j_l})$ with $j_1 < j_2 \ldots < j_l$. The parent of this node has $l - 1$ occupied spares and the module who is not yet failed is i_l. Thus, we deduce that the number of transitions is equal the number of configurations - 1 (for the root). □

3.2.2 *Policy with Spare Reconfiguration*

The second policy produces an acyclic graph. The number of nodes is the same than the one for the previous policy. The number of transitions from an initial configuration is greater and can be computed:

	number of transitions
local	$(\sum_{l=1}^{sm} \sum_{j_l=l}^{sm} f \times C^{l-1}_{f-1} \times A^{l-1}_{j_l-1} \times l(sm - j_l))^c + NB_l$

PROOF. The fact that we accept to reconfigure the spares does not modify the number of configurations. What is changing are the possible paths to reach such a node. For a number l of occupied spares, in the first policy, a child has necessary $l+1$ occupied spares. For this second policy, there are additional horizontal transitions from a node to another with the same number of occupied spares. It is sufficient to compute the number of horizontal transitions and to add this number to the numbers of the first policy. For a node with l occupied spares whose reconfigured modules $i_1, \ldots, i_l$ are allocated on the spares $Conf(i_k) = j_k$ with $j_l \in [l, sm]$. If $j_l < sm$, for $k \in [1, l]$, this node can have an horizontal transition to a node such that $Conf(i_k) = j_l + 1$ and with the same allocation for all other modules. This means that there are $l(sm - j_l)$ horizontal transitions from this node.

We must now count the number of such nodes with l occupied spares and j_l fixed. One application is necessarily on j_l thus there are C^1_f possibilities. There $l - 1$ applications among the last $f - 1$ applications allocated on the $[1, j_l - 1]$ previous spares. This reasoning is the same that for computing the number of nodes. Finally there are $C^1_f \times C^{l-1}_{f-1} \times A^{l-1}_{j_l-1}$ nodes. □

3.2.3 Certification Issue

Certification practices require to perform safety assessment and show real-time predictability for all configurations of a system. Currently, in an IMA-1G platform there is a unique configuration which is completely certified. Because of the large number of reachable configurations (for c=4, bm=5, sm=1 there are 1296 configurations for local reconfiguration and there 146001 for distant reconfiguration) the certification of IMA-2G process must evolve in order to certify a family of configurations. Model based safety assessment, as described in [6], should be able to cope with the large number of reconfigurations. For real-time performances predictability it should be possible to consider that a local module reconfiguration in a cluster has no impact on the performances.

3.3 Operational Reliability Improvement

To assess the operational reliability improvement of IMA-2G over IMA-1G, we compare the causes leading to the loss of an application A that is considered to be a NOGO situation. In an IMA-1G platform, the loss of A is mainly caused by the loss of the basic module hosting this application. In an IMA-2G platform, the loss of A is caused by the loss of the basic module hosting A combined with the failure of the reconfiguration of A that can be caused either because the Supervisor is lost (consequently reconfiguration cannot be performed) or reconfiguration is performed but the spare module hosting A is lost or there is no spare available because other basic modules in the cluster have already failed. In IMA-2G, all causes combine the loss of the basic module with another failure. As the combination size increases the probability of the loss of A decreases and this leads to operational reliability improvement. When we analyze all applications supported by the platform we notice that the loss of the supervisor has an important contribution on the loss probability. To significantly improve operational reliability, it might be necessary to have a redundant supervisor.

4. CONCLUSION

The reconfiguration objectives in the context of the SCARLETT project aims at enhancing the operational reliability of the aircraft, this is somehow a different goal from reconfigurable avionic systems we found in the literature [7, 5, 8, 4, 3] where reconfiguration is one mean to achieve fault tolerance as stated in [7]: "*reconfiguration is one fault tolerance mechanisation for providing expected functionality after a fault*". In the same way, the classical FDIR (Failure Detection Isolation and Recovery) procedure used in most space systems, uses dynamic reconfiguration during the recovery phase. In those cases the system is statically configured with a set of may be redundant (but specialised) equipments which may be powered off/on when failure occurs, in the SCARLETT project we aim at configuring *generic resources*, i.e. IMA modules, with uploadable software functions.

However, even if our primary objectives are different, the methodology, the software and/or hardware architecture design [7, 4, 5], are insightful for our goals:

- the steps of reconfiguration process are usually the same (error diagnostic, select new configuration, apply configuration,...)
- the widespread idea of precomputed and identified configuration as mean to certify configuration seems appealing,
- the need for timing constraint consideration in the reconfiguration process for real-time application.

Furthermore, the reconfiguration principles presented in the ASAAC standard (see [3]) for military aircraft IMA will be of particular interest if we want to explore the distributed implementation of the reconfiguration supervisor. Our approach is close from the one proposed by the authors of [1] although our context is different (they study time triggered systems and dynamic QoS management). The authors propose a *bounded flexibility* solution meaning that the reconfiguration evolves in a bounded and predefined configuration space. This space is reduced to admissible configurations which satisfy the system requirements such as mutual exclusion and fulfil the required QoS level (the system should remain schedulable). We should be able to formalise these configuration space reduction techniques for avionics allocation.

Our mid-term perspectives are the implementation of on ground module and partition level reconfiguration. We also plan to detail the analysis of the proposed solution with respect to operational reliability, safety and certification. A longer term perspective is to study other scenarios including in flight reconfiguration and reconfiguration for safety.

5. REFERENCES

[1] L. Almeida, S. Fischmeister, M. Anand, and I. Lee. A dynamic scheduling approach to designing flexible safety-critical systems. In C. M. Kirsch and R. Wilhelm, editors, *EMSOFT*, pages 67–74. ACM, 2007.

[2] ARINC. ARINC Specification 653: Avionics Application Software Standard Interface, 2005. Aeronautical Radio INC.

[3] ASAAC. ASAAC final draft of proposed guidelines for system issues - volaume 4 : System configuration and reconfiguration, 2004. Aeronautical Radio INC.

[4] S. M. Ellis. Dynamic software reconfiguration for fault-tolerant real-time avionic systems. In *Microprocessors and Microsystems, Proceedings of the 1996 Avionics Conference and Exhibition*, volume 21, issue 1, pages 29–39, July 1997.

[5] C. Metra, A. Ferrari, M. Omana, and A. Pagni. Hardware reconfiguration scheme for high availability systems. In *10th International On-Line Testing Symposium (IOLTS'04)*, page 161, Washington-DC, USA, 2004. IEEE Computer Society.

[6] L. Sagaspe and P. Bieber. Constraint-based design and allocation of shared avionics resources. In *26th AIAA/IEEE Digital Avionics Systems Conference*, 2007.

[7] K. Seeling. Reconfiguration in an integrated avionics design. In *In 15th AIAA/IEEE Digital Avionics Systems Conference*, pages 471–478, Oct 1996.

[8] C. Sriprasad and M. Harvey. Dynamic software reconfiguration using system-level management. In *14th AIAA/IEEE Digital Avionics Systems Conference*, pages 336–, Nov 1995.