

Exploring the Possibilities Towards a Preliminary Safety Case for IMA Blueprints

Graham Jolliffe MSc CEng MRAS; QinetiQ, MoD Boscombe Down; Salisbury, Wiltshire, UK

Dr Mark Nicholson; Department of Computer Science, University of York, Heslington, York, UK

Keywords: Integrated Modular Avionics, Blueprints, Safety

Abstract

The Aim of this paper is to show how a safety argument could be constructed for the use of blueprints in platforms using Integrated Modular Avionics (IMA). It is assumed that the IMA system will contain safety-critical elements. Given current safety analysis techniques, there is no certainty that this can be achieved satisfactorily.

Initially there is a need to define a blueprint, the blueprints will then be considered by looking at their impact on IMA Safety. The ultimate objective of IMA is to produce a reconfigurable system. Whilst this has potential safety benefits, there are substantial problems with the ability to argue that a reconfigurable IMA is safe. Consequently, this paper will concentrate on a 3 Step Approach towards developing full IMA capability. These consist of:

1. Fixed number of prioritised configurations (e.g. lookup table)
2. Ground (static) reconfiguration (between operations)
3. Dynamic reconfiguration

This approach is progressively more complex, but will enable confidence to be gained from success at each step. The safety argument that is produced in this paper is generic and has been produced as part of an MSc project (ref 1). However, the overall IMA safety argument needs to consider many other issues and factors, which may affect the safety of blueprints. This is not covered in this paper, but is expanded in more detail in the project.

Introduction

IMA is a term to describe a distributed real-time computer network aboard an aircraft. This network might consist of a number of computing modules capable of supporting many applications, which in turn may have different safety criticality levels. Most current avionic architectures are federated systems with each function located within its own processor or Line Replacement unit (LRU), connected to each other by an avionics data bus. A typical federated system is shown in the diagram below (Fig 1) (ref 2).

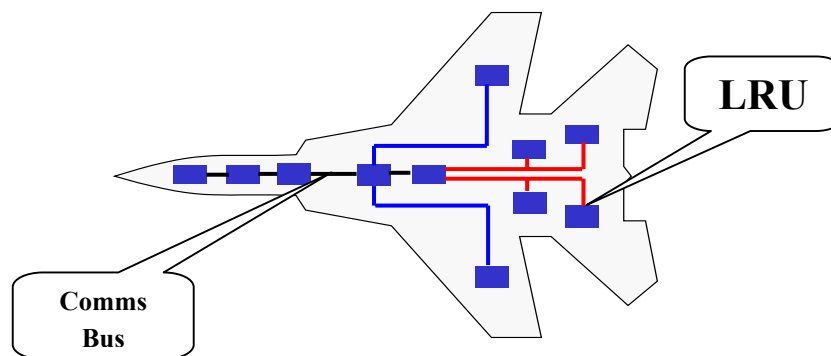


Figure 1 \_ Typical Federated System Architecture

The diagram below (Figure 2) (ref. 2) shows what a typical IMA architecture might look like. Each module contains an application that 'services' either a sensor or output or both. A common shared network connects the sensors, modules and outputs. There are a number of benefits in progressing towards this type of architecture compared with that shown in Figure 1. The civil and military sectors have already clearly identified the benefits of IMA and references 2 - 7 are just a few of the many papers and articles that highlight the advantages of adopting this technology. The IMA concept permits greater flexibility with each function capable of being run as software in any of a number of processors. However, this flexibility is intended to allow reconfiguration of all or part of the systems resources, which has two significant benefits. First, the system can be modified to fulfil different roles or missions.

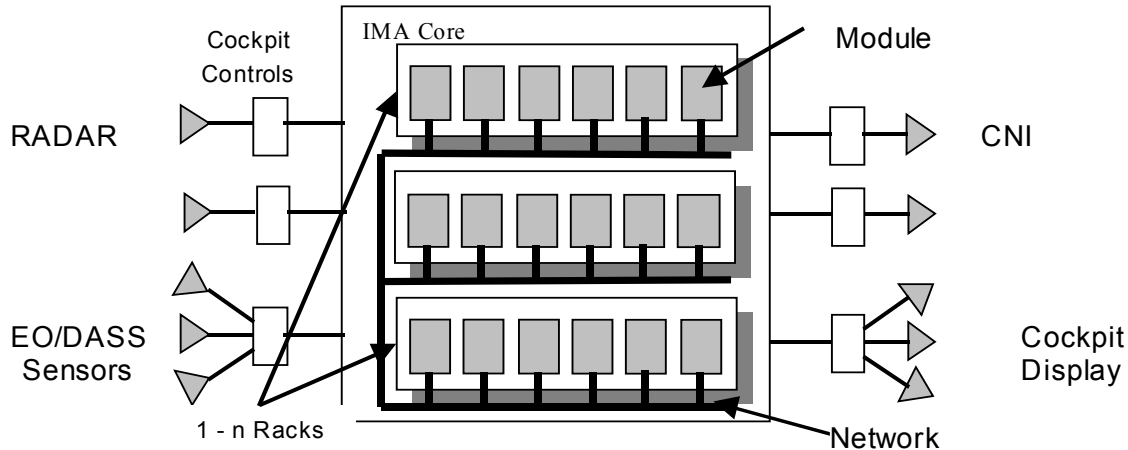


Figure 2 \_ Possible IMA System Architecture

Secondly, the same ability can be used to maintain system safety or integrity, by transferring a safety critical application from a failed component to one that is serviceable. This will also enable more efficient and effective use of processing power and reduce the impact caused by obsolescence. Consequently, IMA promises to be both cheaper and more effective than the existing federated systems. It will also lend itself to a variety of platforms in various environments.

#### IMA Safety

Demonstrating safety is already a complicated task for federated systems, but the number of configuration permutations could be a very large number. If traditional safety analysis approaches were to be utilised for this type of architecture, then the amount of time taken to conduct such analysis would be prohibitive. So an alternative approach needs to be found that will permit safety to be demonstrated adequately, whilst not impeding development and exploitation of IMA systems. A critical issue is the allocation of the applications to hardware in an effective and efficient way whilst meeting safety, reliability and real-time requirements. This has long been realised by most of the agencies and consortia that are considering how best to apply this technology including Eurocae WG60, RTCA SC200 and the Allied Standard Avionics Architecture Council (ASAAC). There are a number of viable methods for implementing IMA, which is currently proposed for a number of new platforms including F22, and Airbus A380. There are also other platforms that claim to be developing IMA systems, but they only partially meet the full objectives and aspirations of IMA as outlined above. Consequently, with so many variations, it might be difficult to prove IMA safety without knowing which 'sort' of IMA is in question.

However, all of the technology groups listed above have the same aspirations for IMA. So, it can be argued that examining the safety aspects of one group should enable read across to the others. This paper will concentrate on the work carried out by the ASAAC. The full aims of the ASAAC programme, and description of the standards and guidelines is not provided in this paper but are outlined in reference 1 and further information can be obtained from references 2 and 8. A number of these groups, including ASAAC, are considering a concept called 'blueprints' as an approach to describing the system. Whilst ASAAC does not specifically propose the use of blueprints, it is generally accepted that blueprints will provide an expedient way of implementing IMA. The safety of a blueprint needs to consider normal operation of the system (including different modes), the blueprint hazards, and how blueprints can contribute to hazards. The contribution of blueprints to safety during system failures also needs consideration.

#### Blueprint Definition

Some work has already been completed on blueprint generation by personnel at QinetiQ Farnborough (refs. 9 and 10), who were consulted on their definition of a blueprint. Similar consultation was made with individuals from industries, who have also been pioneering work within the IMA area. A consensus was reached by a wide selection of individuals representing a good cross section from industry, government agencies, academia and other trusted experts. They broadly concurred with the findings of reference 8 at a meeting held at Farnborough in June 2003. Each blueprint is a generic template for that part of the system, with its own constraints (e.g. hardware performance limitations) from which an optimal solution can be produced, which effectively becomes the System (or Run-Time) Blueprint. The blueprint will contain many features that may not all be utilised within the System blueprint. In effect

it is intended to pick the ‘best’ bits from each blueprint and deliver the System Blueprint that can then be loaded on to the relevant platform. This will depend upon a set of constraints or ‘rules’, which may be either hard (essential) or soft (desirable). The diagram in Figure 3 below shows how this is designed to work conceptually.

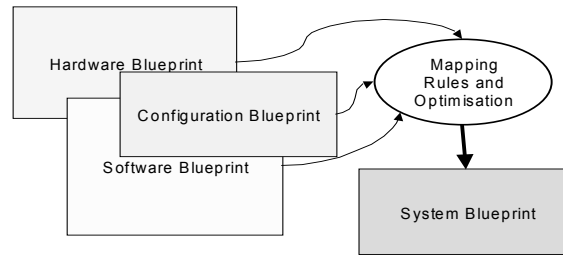


Figure 3 \_ Blueprint Conceptual Relationship

It can be seen that in addition to establishing the safety integrity of each blueprint, the mapping rules and possibly the algorithms used for optimisation must also be evaluated for integrity. As a result of the meetings and discussions mentioned above, the general consensus was that the blueprint model above is a hybrid, consisting of hardware, software and configuration blueprints, as follows:

- a. Software Blueprint – contains a description of each software application in terms of memory requirements (static and dynamic), scheduling requirements, communications requirements.
- b. Hardware Blueprint – contains a description of each type of hardware module in terms of available memory, processor type and speed, and available communications.
- c. Configuration Blueprint – contains a description of how the hardware and applications can be physically and logically connected together e.g. bandwidth, maximum number of connections, etc. This sounds similar to the mapping rules, and could be thought of as a high level ‘filter’. This blueprint predefines how the hardware and software blueprints physically communicate with one another.
- d. System Blueprint – also known as a Run-time Blueprint, is the output from the mapping (optimisation) algorithm, and can implemented on a target system or platform.
- e. Mapping Rules – optimises the Software, Hardware and Configuration blueprints against a set of constraints.

#### Approach and Current Research

In addition to considering the safety of blueprints when the system is performing normally, safety also needs to be assessed during failures, for which the failure modes of a blueprint need to be evaluated. Using the 3-step approach, safety considerations can be investigated, and the safety arguments can then be built up incrementally. Because IMA and the use of blueprints demand an alternative approach to safety analysis, it is easy to conclude that safety will be adversely impacted. However, there may well be positive safety benefits and these will be highlighted.

**Modular Safety Cases:** Modularity has been proposed as a more efficient means of constructing safety cases for complex systems and it would be sensible to examine this work to determine its applicability to IMA blueprint certification. Several papers have been written on this subject including Rushby (ref. 11), which explains that certification is really concerned with abnormal behaviour of systems i.e. if a system can be shown to be fault free then it is safe. However, it is possible that a system could comprise of ‘safe’ components that were not designed to operate together. Such a system would still be unsafe. Even if components are designed to operate together, the ability to show that they are fault free has traditionally been difficult and is not practical for IMA. Rushby (ref. 11) acknowledges that cascade failures are more complex to evaluate, because they may well extend beyond the system module interfaces. A failure can potentially emerge in any other system module that has a dependency. For this approach to be practical, module interdependencies must be kept to a minimum. That may be easier said than done, since some modules will have executive functions that extend to other modules. Therefore, there is a need to conduct a Common Mode/Common Cause analysis as part of the overall IMA safety argument.

Kelly introduces GSN and its benefits as a tool for arguing system safety (ref. 12). It is also argued (ref. 13) that the production of safety cases using Goal Structured Notation (GSN) often results in similar patterns emerging. It makes sense to reuse these patterns that can be used to effectively capture solutions that have evolved over time, building on expertise, and successful certification approaches. However, patterns only avoid duplication of effort. They need to be applied in a particular context in order to be useful in a larger safety case. Kelly (ref. 14) establishes the mechanisms for managing and representing safety cases as a composition of safety case modules as described below.

- a. Objectives addressed by the module
  - b. Evidence presented within the module
  - c. Context defined within the module
  - d. Arguments requiring support from other modules
- Inter-module dependencies:
- e. Reliance on objectives addressed elsewhere
  - f. Reliance on evidence presented elsewhere
  - g. Reliance on context presented elsewhere.

The relationship between modules can be complex, with supporting evidence being derived from multiple sources. Therefore, it becomes important to understand the objectives, evidence and context defined for each module, since this could jeopardise a safety case argument if there are inconsistencies. The support from other modules and the reliance needed to provide that support should not contradict the first three items in the above list. Extensions to GSN (ref 12) are introduced to enable the concept of Safety Case ‘Modules’ to be represented. The concept of ‘Away Goals’ is also introduced, which forms part of the argument for a particular safety case that requires the support of another module. Consequently, the relationship between various modules has to be well defined. It is recommended that an ‘away goal’ always be first satisfied by a single module. Only if this is impracticable, should combinations of sub-goals from multiple modules be used. However, in order for these modules to be integrated into a larger safety case or argument, some rules need to be established. For instance, it would not be acceptable to use an ALARP pattern as a solution to a Safety Margin Pattern because the safety margin is likely to be a fixed target. Whereas the target for ALARP will vary according to a number of system criteria, i.e. the two arguments have goals that do not match.

**Safety Case Contracts:** Kelly (ref. 12) recommends that once a successful match has been achieved, this should be recorded as a ‘contract’. A suggested format for such a contract is shown below in Table 1.

Table 1 - Proposed Format for Safety Case Contracts

| <b>Safety Case Module Contract</b>   |                      |                        |               |
|--|----------------------|------------------------|---------------|
| Participant Modules (e.g. Modules A, B, C, etc.)   |                      |                        |               |
| <b>Goals matched between Participant Modules</b>   |                      |                        |               |
| <i>Goal</i>  | <i>Required By</i>   | <i>Addressed By</i>    | <i>Goal</i>   |
| (e.g. Goal 1)  | (e.g. Module A)      | (e.g. Module B)        | (e.g. Goal 2) |
| <b>Collective Context and Evidence to be held Consistent between Participating Modules</b> |                      |                        |               |
| <i>Context</i>   |                      | <i>Evidence</i>        |               |
| (e.g. Context C1, Assumption A2, etc.)   |                      | (e.g. Sn1, Sn 3, etc.) |               |
| <b>Resolved away Goal, Context and Solution References between Participating Modules</b>   |                      |                        |               |
| <i>Cross-Referenced Item</i>   | <i>Source Module</i> | <i>Sink Module</i>     |               |
| (e.g. Away Goal AG3)   | (e.g. Module B)      | (e.g. Module A)        |               |

This should include the participants and the relevant goals, context and evidence that each participant brings to the contract. How a contract-based approach can be used has been proposed at reference 15, which argues that sensibly chosen modularity can considerably benefit a system when under change. This should be beneficial for system changes due to upgrade, replacement through obsolescence and reconfiguration, all of which are objectives of IMA systems. The contract format proposed by Kelly above is a good starting point in terms of broad safety integrity objectives. However, this information would only form part of the overall data needed to enable reconfiguration to occur. For instance, we have already determined that performance and resource requirements will be needed. There also needs to be some form of weighting or prioritisation allocated to safety information and performance information in order to determine how best to optimise the system.

#### IMA Methods of Reconfiguration

As previously described, this paper covers an incremental approach to arguing the safety of reconfiguration in an IMA system commencing with manual reconfiguration, followed by static automated reconfiguration, and finally, dynamic

automated reconfiguration. This approach will enable safety experience to be gained through the increasing levels of reconfiguration complexity. Each of these methods is described below in more detail, but it is emphasised that these descriptions are not definitive, and each method may have variations in reality.

**Manual Reconfiguration:** This will consist of a limited number of predetermined configuration options. In its most basic form, a new configuration is chosen that best suits the intended use by the operator. However, variations of this method might include the possibility of limited automatic reconfiguration depending on certain faults. A decision making process that selects which configuration is best to use is required. This may be relatively straightforward for a small number of configurations, but may need the use of procedures for larger numbers of configurations. So, for instance if there is one configuration for each role of the platform, then it is simply a matter of choosing the configuration that matches that role. For more complex configurations, the operator might have to follow a flow chart that asks a number of questions that lead him to the best configuration depending on the answers supplied.

**Static (Ground) Automated Reconfiguration:** This is the next stage in the 'incremental' approach to IMA reconfiguration and represents an increase in the number of possible configurations that is too large to select manually as described above, hence, the need for automation. This brings a number of additional safety problems to resolve. First, it will be impossible to analyse all of the possible configuration permutations; therefore, it may not be possible to positively determine that the product (configuration) is error free. Consequently, there needs to be a combination of evidence showing that the process of reaching a particular configuration is error free, coupled with any testing of the configuration that can be achieved prior to use.

Secondly, a choice as to which optimisation algorithm to use has to be made. From the algorithm review (ref. 1) and the conclusions of reference 9, it would appear that Simulated Annealing has a consensus in its favour. However, the rationale for this conclusion is not particularly strong, and is dependent on its benefits for handling hard and soft rules. Another factor is the amount of time available to perform the optimisation process, which in turn has to be balanced with the need to conduct some testing of the chosen configuration.

**Dynamic Reconfiguration:** Dynamic reconfiguration is the final step in the proposed phased approach, and will be the most difficult version to argue that sufficient safety assurance can be achieved. This method of reconfiguration has all of the characteristics that the Static reconfiguration method offers, but the intent is that this can be achieved whilst the aircraft is operating. The technique will enable a change in the role of the aircraft in flight, thus enabling the aircraft to complete different missions without the need to return to base. Similarly, the aircraft will be able to automatically substitute for systems that have failed. This has the double benefit of maintaining some mission capability, though probably reduced, and adding another layer of safety. However, these two benefits could be applied at each other's expense, so a balance of priorities will be needed. Although these benefits are clearly advantageous, there are a number of problems that need to be addressed. These are discussed more fully (ref. 1), but include the current lack of processing power to run the optimisation algorithms quickly enough to enable reconfiguration to occur in real time. However, for the purposes of this paper it will be assumed that this technical obstacle has been overcome, so that the numerous safety issues can be discussed and subsequently addressed. Dynamic reconfiguration also has to take into account the need for a safe transition between configuration states. One other safety benefit of Dynamic reconfiguration is the possibility of fault reporting. If a system has detected a component failure that warrants a system reconfiguration, it should be able to record the information to enable maintainers to quickly locate and replace the failed component when the aircraft returns to base.

#### Other IMA Safety Issues

There are a number of IMA safety issues that do not directly affect IMA Blueprint safety. These are covered in some detail (ref 1), but are simply listed below in order to emphasise that Blueprints are not the only IMA safety issue. They include: Priority Deconfliction, Platform Roles (Peacetime or Wartime Scenarios), Operating System integrity, Communications integrity, Processor integrity, and there are also existing Safety Case constraints. The Safety Case will usually contain argument and evidence that demonstrate the safety of a system. This will include evidence from the Hazard Log and Hazard Tracking System, but is also concerned with other risks associated with the product including compliance failure with standards or contract terms. Reference 16 contains a suggested list of contents that might be expected to make up a typical safety case. Missing from this list is the overall safety argument, which should draw upon the individual pieces of evidence to produce a convincing rationale that the system is demonstrably safe.

The safety case also needs to be updated to reflect changes to the system, the problem for arguing reconfigurable IMA safety, is that the system is potentially always in a state of continual (or at least regular) change. Current safety

analysis techniques can deal with this as long as the changes are predictable. Without this predictability, it will be impossible to present a reasoned safety argument for IMA. The reality is that the configurations that the IMA system can attain should be deterministic. However, the potential for IMA means that the number of configurations is potentially huge. Thus preventing timely or expedient analysis of the complete configuration set. Given this situation a number of potential solutions are discussed later for both the static and dynamic reconfiguration scenarios.

### Progressing an IMA Blueprint Safety Argument

**Technical Openness:** The need for technical information to be made readily available in order to provide some agreed measure of integrity is well recognised, especially if modularity is to be an accepted means of developing Safety Cases for IMA systems. Such modules will require the use of safety contracts to enable goals to both be supported and requested depending on where the module resides in the overall safety argument. Therefore, if a subordinate module is offering support for a higher module, it needs to be able to express a means of providing that support. Ultimately this will take the form of evidence, which will include technical information where necessary.

However, even without the need for this information in support of the safety case, a lot of information will be required by the operating system to enable configuration optimisation and reconfiguration to occur. For example, a whole range of metrics will be needed to enable the optimisation algorithms to function correctly. In addition, this information will have to be provided in a common format that is readily understood by the algorithms. Currently, the need to provide this information is restricted to only those components that require access to each other. For IMA, however, this information needs to be available to the whole system in order that the system, through using the algorithms, can determine how it should reconfigure itself. This is potentially an area of very high risk for IMA, because existing federated systems are bound in a web of contracts involving, design authorities, suppliers, customers, certification agencies, auditors and independent assessors. Without a free exchange of information, it will be difficult to make an argument in favour of its safety, and there will certainly be insurmountable technical difficulties.

**Fault Investigation:** The need to correctly identify faults is paramount to the overall safety argument for blueprints, and hardware blueprints in particular. The implication is that in addition to being able to reconfigure itself to the next optimal configuration solution, the system should also have the ability to identify, and record the nature of the fault.

**Search Algorithms:** The ability to reconfigure these systems to enhance performance, mission capability and safety will be very restricted unless a means can be found to enable optimised configuration solutions to be quickly identified and implemented. Such processes need to be automated in order to cope with the complex factors that require consideration, and achieve an optimal solution within very limited time constraints. There are a number of algorithms that can be used for searching for optimal solutions of configurations. Each has different characteristics and, therefore, each has different advantages and disadvantages. The properties of some of these algorithms have been examined (ref. 1) to establish their relative merit in terms of demonstrating integrity.

**Measurement of Optimisation:** This is essential for determining if an optimal solution has been found, but it can also be a method of ensuring that minimum safety requirements have been met. So, for instance if a search reaches a point where a required set of conditions has been met, then it can terminate. To enable this, it is necessary to measure the value of each solution in some way, in order to compare it with the required value. Many attributes will need to be assigned values, which act as a means of prioritisation. These attributes or constraints need to be satisfied and this could be achieved by setting the optimisation algorithm some constraint targets to be met. So for instance if safety and performance are two such constraints, then the target for a peacetime mission scenario might have a high safety target, and lower performance target. This also assumes that both safety and performance can be measured to compare against the target set. In reality, providing such metrics is problematic, particularly for software.

### IMA Blueprint Safety

This section takes those issues that are pertinent to IMA blueprint safety and attempt to construct a generic safety argument to support the use of IMA blueprints. As previously discussed the potential for complexity with IMA systems is likely to overwhelm the traditional approach taken for federated systems. However, it should be possible to show what a safety case should look like, or at least how safety for IMA can be argued. The approach taken below uses hazard analysis to determine the required integrity level of each IMA component using the ASAAC model as a basis. This information can then be used to determine what safety evidence is required.

**Top-Level GSN Safety Argument:** The GSN diagram below (fig 4) attempts to show the safety argument for IMA in each of the three methods of implementation. This is a top-down approach to arguing safety for IMA that is developed in much more detail at reference 1. There are a number of key elements that will determine ultimately

whether a total safety argument can be constructed. However, not all goals need to be achieved for each method, G8 is only needed for dynamic reconfiguration and G4 is only needed for static and dynamic reconfiguration.

Of the context boxes C1-3, C2 is likely to be the most awkward to fulfil, because the IMA system cannot properly be defined until its configuration is known. However, there will be a certain amount of system knowledge available, including the role requirements, which should be able to assist with identifying underlying system safety requirements. For instance, there will be certain safety issues on an aircraft, which will apply regardless of the actual system configuration used. An example might be that the undercarriage is not to be retracted when the aircraft has landed. However, care still has to be taken even with something as obvious as this if the aircraft in question is amphibious.

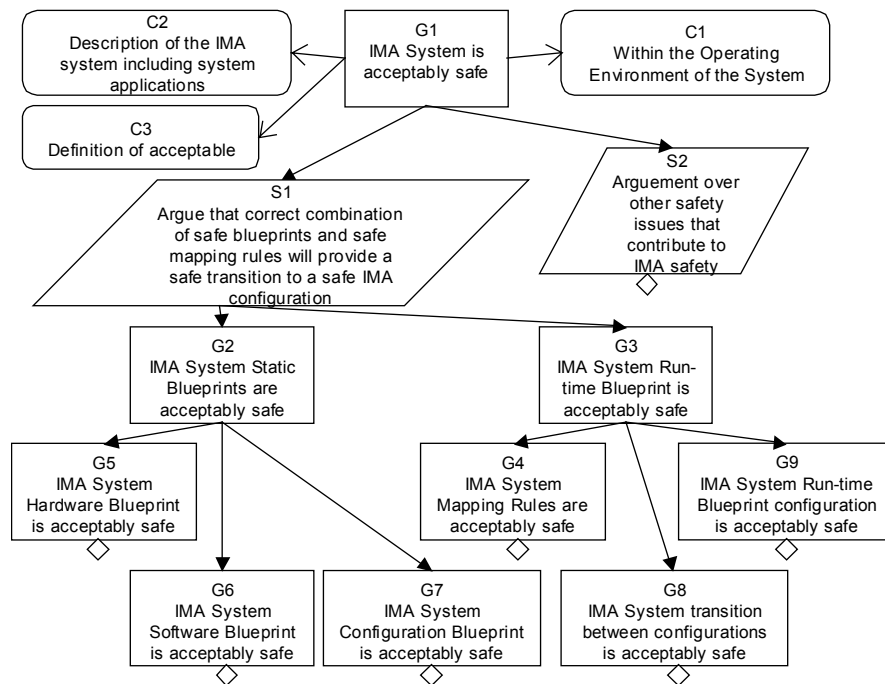


Figure 4 \_ Top-Level GSN Safety Argument for IMA

These gaps in system knowledge can be filled by information supplied and assumptions made for individual components of the static blueprints under goal G2. The strategies S1 and S2 are key to a successful overall safety argument. S2 is included to capture the issues that are not pertinent to IMA blueprint safety. S1 is based on what safety evidence is likely to be available. It is theoretically possible to produce product safety evidence that incontrovertibly demonstrates the safety of a particular configuration. This is the basis for current system certification. However, it is impractical to cover all potential configurations in this way, so S1 attempts to argue that there is sufficient evidence from the blueprints, the mapping rules and transition between configurations to demonstrate IMA configuration safety. These goals are shown at G2 and G3, where the latter is likely to be the most difficult to satisfy, because of the unknown properties of optimisation. If it can be assumed that any reconfiguration on the ground is safe, then G4 is only applicable to dynamic reconfiguration.

The most difficult of the remaining goals is G8, which covers the safety of the Run-Time blueprint. It will certainly not be sufficient to argue that the process integrity for determining the Run-Time blueprint is, by itself, sufficient evidence of safety. Apart from anything else, the process is far too complex to support an argument based upon process evidence alone. Some form of product evidence is also required and the development of this goal (ref. 1) makes some recommendations as to how this can be achieved.

**Safety Aspects of Blueprints:** Looking more closely at the safety aspects of blueprints, particular attention to their safety is needed when the system is in one of the following modes: Power-up, operating normally, faulty, shut down and changes state (between configurations). However, these aspects do not apply to all of the blueprints defined. In particular the Software, Hardware and Configuration blueprints are effectively fixed at any one point in time, since they represent availability of resources. However, the Run-Time blueprint will need to consider all of the above

conditions. Of course if the software is modified, or the hardware updated, then they too may change, but that does not need to be taken into consideration when assessing integrity at one moment in time. Therefore, it is assumed that none of these blueprints will change state, though we do need to demonstrate that faults do not propagate and invalidate the required integrity of the Run-time blueprint. These 'Static' blueprints will still need to be able to demonstrate their safety properties. This is not only to enable certification agencies to approve their use, but perhaps more importantly, to enable the safety properties to be used by the optimisation algorithms, through which the correct level of system integrity can be maintained. Before we can attempt to develop a safety argument for blueprints, there needs to be an analysis of the hazards associated with blueprints.

The HAZOP technique was used for each of the four types of blueprint and is detailed at reference 1. The HAZOP technique has been chosen because it is a structured method of hazard analysis, which is less likely to overlook potential hazards and a preferred technique of the MoD (ref. 17). The choice of guidewords has been determined by those recommended by reference 18. It is not feasible to guess what components might be included in any one blueprint, so this analysis is based upon generic hardware, software and configuration components. These can be treated as separate entities for these blueprints, but the Run-Time Blueprint will also have to consider how the components from each of the other blueprints might combine with one another. Again, this will be treated generically at this stage, since the number of permutations will make a full analysis impractical for this paper.

**Blueprint Context:** The generic Safety Case Patterns (ref. 12) can be used in a variety of domains; however, all of them require some knowledge of the system hazards and/or system description. Unfortunately, such a description will not be known until it has been configured, so a Top-Down approach would not appear to be of much benefit at this stage. It also implies that ALARP is not a sound method for arguing IMA safety. From a system perspective, there are three main reasons for not using ALARP. First, there is the problem of identifying system hazards. Second, there is the problem of trying to achieve the optimal solution, when an acceptable solution is all that is required. Finally, it will be impractical to assign a proportion of safety to each component, when it may not be possible to determine how that component is likely to be used in any given configuration. For instance, it may be used in isolation, in parallel or in series with other components, or it may even be used to enable system redundancy.

This paints a somewhat pessimistic picture of the ability to define operating context, but there are ways of overcoming this. There is an argument that the choice of mapping rules can be used to define the operating context. Another approach to this problem is to consider a component of a blueprint that has been designed with a particular system-operating environment in mind. If such information or evidence is available then the gap between the top-level context C2 in Figure 4 and the blueprint safety argument can be bridged.

**Blueprint Safety Claims:** If this gap cannot be bridged then an alternative Bottom-Up approach may be required, because it can make use of safety claims. Now as Kelly (ref 12), points out, this can only be used as supporting evidence, in other words, there needs to be some form of over arching argument that can be used to demonstrate blueprint integrity. Kelly has also identified General Construction Safety Case Patterns that include Diverse Argument and Safety Margin. Both have potential for arguing blueprint safety. However, of the two, the diverse argument will probably be the more difficult to prove in reality, partially due to the difficulty of defining diversity and also because it is notoriously difficult to demonstrate the complete absence of common mode failures. Hence, the need to include Common Mode/Cause Analysis as discussed earlier.

The Safety Margin pattern has much greater potential for IMA. First, each component of the blueprint in question can be shown to have a 'safety margin'. Secondly, that safety margin can then be used by the optimisation algorithms to achieve overall system safety requirements. However, the weakness with this approach is that individual components may only be just safe enough in a given operating context. Outside of that context, there may be insufficient safety for that same component. Even if it remains just safe, there may still be a problem if the overall safety margin cannot be achieved. It is pointed out (ref. 12) that there is a danger of over-engineering if there is an excessive safety margin, which traditionally has been resolved by applying experienced judgement. That is more difficult to achieve with optimisation algorithms because as yet, there is little by way of experience to form such a judgement.

Although safety margins are relatively easy to apply with only one criterion, IMA uses a mixture of hardware and software components, whose means of measuring safety differ substantially. The means of connecting this mixture of components will also have a bearing on the overall safety integrity, because some are quantitative and others are qualitative. All four parts of the blueprint process will be looked at since each will impact on the overall integrity of the IMA system. However, ultimately it is the Run-Time blueprint that has to be demonstrably safe. Conceivably, there may be elements of the software and hardware blueprints that are not safe individually, but when combined in a

particular configuration by the mapping rules, provide sufficient integrity. This will be explored later, but as already stated this project is looking at IMA in three steps, and it is simplest to consider the manual reconfiguration case first.

Phase 1 Manual Reconfiguration: Each of the static blueprints was assessed from a safety argument perspective, using simple generic models. The models and the detail of these blueprint arguments can be found at reference. 1. The hardware blueprint was examined at power-up, under normal operating conditions, at power down and under failure. The safety of the latter is considered most important and it was shown that the ability to identify faults is critical, thus requiring high integrity health monitoring. Maintaining an 'image' of the system configuration was also seen as a necessary requirement, in order to act as a reference point for any further reconfiguration and may also assist with subsequent fault rectification. A detailed generic HAZOP has been produced for each of the static blueprints (ref. 1), demonstrating how Hazards relating to these blueprints can be identified and assessed. The results can then be used to determine which goals are needed in the safety argument to address the hazards. The hardware HAZOP assumes that each (safety critical) component receives information and power and outputs data. Note that the processing of the information will be carried out by a suitable component in the Software blueprint, and the transfer of that information (or data) will be controlled by a suitable component in the Configuration Blueprint.

The examination of the software blueprint identified that specific safety targets could not be applied. However, in order to produce a rigorous safety argument, it is necessary to ensure that the development and testing processes are of the highest integrity. This includes the need for independent verification and validation. Although this is no different from the requirements for current high integrity software, it should be appreciated that IMA is likely to require greater dependence on high integrity software components to allow greater reconfiguration flexibility. It should be appreciated that software reconfiguration should only occur where the reconfigured software is of a similar or better integrity, unless it can be shown that not reconfiguring degrades integrity further.

The configuration blueprint safety argument identified the need for safety contracts, which should be completed as part of the initial system safety analysis and updated as the system is changed. However, there is a need to ensure that the optimisation algorithms can utilise this contract information via an information model. This should be considered for further research. Fundamentally, the Runtime blueprint more than any others has to be proven to be safe. The preceding blueprints can all provide evidence of safety that can help with the overall safety argument, but the Runtime blueprint is what will determine the actual software configuration on the aircraft. However, for manual reconfiguration, it can be assumed that the system is safe prior to and post reconfiguration, since the system will be unpowered. A go/no go test will be required post reconfiguration, but the only other safety consideration for the manually reconfigured run-time blueprint is that any rigs or equipment used are also safe.

Similarly, for the Mapping Rules, the manually reconfigured system is not truly being optimised. Instead, a configuration is chosen from a limited number of predefined and pre-evaluated configurations. The main issue here is that the evaluation of each of the configurations will require considerable effort. This effort could be made more efficient with the use of modular safety cases and the use of safety contracts.

Phase 2 Static Reconfiguration: Since the Hardware, Software and Configuration Blueprints can be assumed to remain the same as for the Manual/ground reconfiguration case above; there will be no additional safety requirements for the Static Reconfiguration case. The principal additional safety consideration for the run-time blueprint is that the reconfigured system needs to be demonstrably safe. This is primarily because there are many more permutations of configuration compared with the manual reconfiguration scenario that cannot be pre-assessed for integrity. Post reconfiguration testing and a 'sanity' checking are recommended, however, neither can be considered foolproof.

Some additional confidence can be gained by comparing a reconfigured system with known configuration patterns that have been proven to be safe. If a particular configuration can be shown safe, then variations of that configuration could have implied safety characteristics if the variation provided a safety benefit. As long as the differences can be shown to be either similar or to improve safety, then an 'at least as safe as' argument can be made that the new configuration is also safe. Nevertheless, the run-time blueprint safety argument is still relatively weak and needs evidence of integrity from the optimisation process to provide additional reassurance of overall integrity.

The choice of algorithm must be made based on a number of factors including; the type of problem; the size of the problem; computing power available and the importance of optimal against near optimal solutions. For example, it may be quite acceptable to produce a near optimal solution as long as mission requirements are met and safety requirements are not compromised. These are called System Design Factors (SDFs) (ref. 19). Once these have been identified, it is then possible to address other issues such as the granularity of measures used for evaluation, and the speed at which the evaluation is to occur.

**Phase 3 Dynamic Reconfiguration:** Although some limited automatic reconfiguration is technically achievable now, the computational power to permit full system reconfiguration is still some way off. Unlike the scenarios for the previous phases, dynamic reconfiguration has to take account of the initial system state, the transition between states, the correctness of fault detecting and reporting and the selection of appropriate an optimisation algorithm. Fault reporting has already been mentioned under the hardware blueprint summary above and the initial system state can be assumed safe prior to the need for reconfiguration. A further factor is the need to complete the reconfiguration in real time; thus, optimisation time is now a constraint. However, this imposition makes it more likely that an optimal solution will take too long to identify, and there is a higher risk that a solution may not even be acceptable.

The transition between states may be achieved using a maintained 'image' of the initial configuration, such that system settings, data values, etc can be synchronised before transition occurs. However, an incorrect reconfiguration is more problematic and can occur for two reasons. First, a hardware fault could occur, which should be detectable, and thus prompt a subsequent reconfiguration. The second reason is result of an algorithm error leading to an incorrect reconfiguration. This is much more difficult to detect, unless the error is obvious to the operator. Clearly, much depends upon the integrity of the optimisation algorithm. The selection of the optimisation algorithm is made more difficult due to the time constraint imposed by the real time environment. This will lead to compromise solutions, and it could be deduced that the potential for erroneous solutions will increase as a result.

Although there is a very real potential improvement in safety through the use of reconfiguration, it needs to be argued and correctly prioritised. In summary, of the four additional safety criteria that need to be considered for Dynamic reconfiguration listed above, only the safety of the initial state is without any major problems to resolve. Apart from further research in this area, the best means of achieving integrity is probably through the phased approach proposed in this report. As confidence in the use of these algorithms grows through the previous phases, so an argument can be built based upon previous experience. Currently, however, this remains a weak argument. In the meantime, the technical difficulties of implementing dynamic reconfiguration can be advanced by the use of look-up tables, which is akin to a semi-automatic Ground Reconfiguration condition.

#### Conclusion of IMA Blueprint Safety Progress

The aim of this paper was to explore the possibilities of developing safety cases for IMA blueprints. The ultimate objective of dynamic reconfiguration is difficult to achieve technically and difficult to determine integrity. Therefore, it was proposed to adopt a three phased approach. For each phase, each of the various blueprints defined were subjected to a hazard analysis, using the HAZOP process. From there, a safety argument can be produced for each.

**Conclusion of Overall IMA Safety Issues:** A number of alternative approaches to IMA safety were discussed which include Safety in Concept and Technical Openness. Safety in Concept is advancing the notion that if safety can be incorporated at the design stage, then it should follow that it can be incorporated within the conceptual stage as well. This requires further thought and consideration, but the fact that IMA can potentially be fault tolerant already demonstrates that safety is part of the system concept.

Technical Openness stresses the need for sharing information. This is an ongoing requirement for evaluating system safety currently, but for IMA, the need to share information will be essential to not only determine system integrity, but is likely to be necessary to achieve technical success as well. Although this is a very high-risk issue for IMA, no recommendations have been made at this time. This is because each stakeholder has a personal stake in overcoming the current problems regarding the withholding of information. Therefore, they will need to work together to solve this problem, but it would be inappropriate to provide advice on how this should be achieved, since much will depend upon the relationships developed between individual suppliers and their customers. However, efforts are already being made to address this, which should resolve this issue.

#### References

1. Jolliffe G (2004). Exploring the Possibilities Towards a Preliminary Safety Case for IMA Blueprints, MSc Project, Department of Computer Science, University of York, 2004.
2. Kemp J (2000). ASAAC – An Overview Issue: 01, 2000.
3. Conmy P (2003). P Conmy's Home page, [http://www.users.cs.york.ac.uk/~philippa/IMA\\_LINKS.html](http://www.users.cs.york.ac.uk/~philippa/IMA_LINKS.html) , 2003.
4. Tudor N (2002). Realising Integrated Modular Avionics In Military Aircraft, 2002.
5. Aviation Today Magazine Website, [http://www.aviationtoday.com/cgi/av/show\\_mag.cgi?pub=av](http://www.aviationtoday.com/cgi/av/show_mag.cgi?pub=av), 2003.

6. MoD ADAS(Air) Aviation Support Vision Website, [www.ams.mod.uk/ams/content/docs/fse/fse-avs/inmodavs.htm](http://www.ams.mod.uk/ams/content/docs/fse/fse-avs/inmodavs.htm), 2002.
7. Thales' A380 website <http://www.thalesgroup.com/all/pdf/a380avionics.pdf>, 2003.
8. ASAAC Phase II Stage 1 - Stage 1 Executive Summary, 1999.
9. Stevens B (2002). IMA Configuration – Preliminary Blueprint Description, QinetiQ Report, 2002.
10. Murray T (2002). Specification for a Run-Time Blueprint Generator, QinetiQ Report, 2002.
11. Rushby J (2002). Modular Certification – CSL technical report, June 2002.
12. Kelly T P (1998). Arguing Safety – A Systematic Approach to Managing Safety Cases, Department of Computer Science, University of York, 1998.
13. Kelly T P, McDermid J A (1998). Safety Case Patterns - Reusing Successful Arguments, In Proceedings of IEE Colloquium on Understanding Patterns and Their Application to System Engineering, 1998.
14. Kelly T P (2001). Concepts and Principles of Compositional Safety Case Construction, Department of Computer Science, University of York, 2001.
15. Bates S et al (2003). Safety Case Architectures to Complement a Contract-Based Approach to Designing Safe Systems, Proceedings of 21st International System Safety Conference, Chicago, 2003.
16. Storey N (1996). Safety Critical Computer Systems, Addison Wesley 1996.
17. Defence Standard 00-56 Safety Management Requirements for Defence Systems (Pt 1) Iss 2, 1996.
18. Defence Standard 00-58 HAZOP Studies on Systems Containing Programmable Electronics Iss 2, 2000.
19. Nicholson M (1998). Selecting a Topology for Safety-Critical Real-Time Control Systems, Department of Computer Science, University of York, 1998.