

# Generating and Maintaining a Safety Argument for Integrated Modular Systems

Mark Nicholson, Philippa Conmy, Iain Bate, John McDermid  
Department of Computer Science, University of York, York, YO10 5DD, UK  
Email: mark | philippa | ijb | jam @cs.york.ac.uk  
Tel: (+44) 1904 432789

## Abstract

*The aerospace industry has been investigating integrated modular systems (IMS) for some years. These systems offer benefits in terms of flexibility, software/hardware abstraction, and incremental upgrades. However, in order to benefit from the technology a safety case must be generated which can be maintained incrementally with system changes, otherwise certification will be prohibitively expensive. This paper presents a baseline safety case for IMS in which evidence can be separated between different stakeholders in the system. The different types of incremental upgrade are then considered and a method is proposed for determining the impact of the upgrade on the baseline safety case and elements of the IMS.*

## INTRODUCTION

Conventional aircraft systems are *federated*, with each major function, or application, in a separate hardware unit. These units may be interconnected, but each is essentially considered independently from the point of view of certification. Federated systems can be seen in most current aircraft and are expensive to develop and certify. These systems generally have software which is tightly coupled to the underlying hardware platform. They are relatively inflexible in that small functional changes can give rise to significant amounts of rework to the certification basis of the aircraft. Platform changes necessitated by hardware obsolescence can also lead to considerable effort and rework to re-certify. Even if a batched change mid life update (MLU) approach is employed the difficulty of analysing the effects on the certification basis, and other factors such as loss of knowledge about the design, means that a completely new safety case is often produced. Thus, much of the previous information is either lost or work undertaken that is not required.

These limitations, amongst others, have led to

research into an alternative approach known as *Integrated Modular Systems (IMS)*. In the aircraft industry the term commonly used for an IMS is Integrated Modular Avionics (IMA). We believe, the comments made in this paper are relevant for non aircraft sector specific systems as well, and therefore the term IMS is used.

The aim of IMS is to bring the flexibility of distributed architectures, such as networks of PCs, to aircraft applications. With IMS a number of functions or applications run on a processor, communicating via services provided by an operating system. IMS has many potential benefits including simplifying software upgrades, making it feasible to add new applications without recertifying the whole IMS, and assisting in meeting requirements for maintenance free operating periods (MFOPs). The overarching aim is to reduce the cost of developing and maintaining a system through its lifetime.

The use of IMS changes traditional system architectures in a number of ways. For example, there will be no physical boundaries between applications. A logical separation approach, called partitioning, is employed instead. Also, sensors and actuators may be interfaced to buses via remote data concentrators, rather than linked directly to the hardware unit where they are used.

However, in this paper, our main focus is on the impact that the introduction of such Integrated Modular Systems has on *safety certification* and the maintenance of a safety argument through the lifetime of an IMS. Thus in Section 2 the current state of the certification process for avionics systems is considered and in Section 3 we introduce the structure of a safety argument for an IMS. The IMS chosen is based on the ARINC 651 (ARINC 1999) standard. Emphasis is placed in this section on the ability to separate the arguments relating to the core services provided by the IMS and the applications that reside on the IMS.

In Section 4 investigates why such a separation is vital. Firstly, to facilitate incremental certification. Incremental certification is perceived to be one of the

major advantages of moving to an IMS as it provides the ability to integrate and qualify new applications, and maintain existing applications, without the need to re-qualify the whole platform. Secondly, to limit the impact of future modifications so that as far as possible only the modified parts need to be re-certified. The concept of a set of "equivalent" systems (Nicholson, Hollow et al. 2000), and system safety arguments, is used as a unifying theme for the work on incremental certification.

## 2. CERTIFICATION OF AVIONICS SYSTEMS

Under current certification procedures each system is assessed in isolation, and the certification process assumes that the software behaves in a deterministic manner. Recent international standards such as ARP-4754 (SAE 1996) and the accompanying ARP-4761 are intended to deal with "complex and integrated systems" for commercial aircraft. However they do not explicitly deal with issues such as IMS and, implicitly, they still reflect the "system at a time" approach to certification. Military standards such as DS 00-55 (MoD 1997) and DS 00-56 (MoD 1996) in the UK, and MilStd 882C (DoD 1996) in the USA are similarly mute on the subject of certification of IMS.

Bradley (Bradley, J. et al. 1996) lists seven areas of avionics system certification which are affected significantly by the use of IMS principles and technology:

- Isolation can no longer purely be provided by physically separating the system functions.
- Non-cyclic scheduling, such as priority based scheduling (Burns 1995), will be required to support varying workloads (this is non-deterministic, although it is predictable) and to allow the worst case timing characteristics of the system to be determined.
- Common cause failures may be introduced by means of the management units employed to provide isolation and reconfiguration.
- Safety critical application functions will be placed on standard commercial processors.
- Hardware modules will need to be interchangeable for ease of maintenance.
- Reconfiguration can affect system safety analyses, e.g. zonal analyses, as well as just properties local to a "system".
- If systems are to be allowed to evolve, then it must be possible to re-use certification evidence for those parts of a system that have not changed – even though the old and the new functions may

share resources – otherwise certification will be prohibitively expensive.

In this paper we concentrate on the last of Bradley's issues. This boils down to providing a safety argument and maintaining it throughout the lifetime of the system via incremental certification. For example, modifications are considered in section 11 of ARP 4754. However, this section does not explicitly address the implications of incremental update, including the need to maintain the safety argument through the lifetime of the IMS. Thus, the standards do not help when considering IMS. There is a need to go back to more basic principles. We therefore introduce the basis of a safety argument for an example IMS.

## 3. "BASELINE" SAFETY ARGUMENT FOR AN IMS

### Computing Equipment and Interactions in IMS

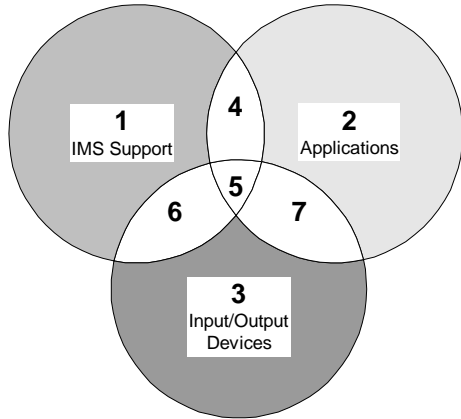
One of the issues surrounding IMS that makes safety argument production and maintenance difficult is that the exact form of IMS to be employed in complex computer based control systems has not yet been resolved. At a recent workshop on IMS the number one question on the brainstorm list was "*What is IMS?*"? Therefore, we have chosen one particular exemplar that has the advantage of being based on a standard, ARINC 653.

In this case IMS contains a number of computing modules. There are groups of modules within IMS cabinets, which are positioned throughout the platform. These are linked via a computer network to each other and to various input/output (IO) devices such as sensors and actuators. Within each module there is at least one partition containing application data to run on that module. A partition is an area logically separated from other application areas and the operating system, both for scheduling purposes and to protect data/code memory space.

The partitions on a module are scheduled in a cyclic manner, allowing each to access the core processor. Within the allotted time slot there is another schedule controlling individual processes in the partition. An application may be divided into more than one partition.

The computing elements can be divided into three sets. These are application elements (e.g. specialised computing), the IMS computing supporting elements (e.g. processors, operating system), and input/output devices (e.g. sensors/actuators, display devices). These elements are not physically isolated within the system, but are logically mapped; for example applications are

data-loaded into partitions on the IMS bare platform (BP) modules. The intersections of the sets represent the various (logical) interactions of the elements. The sets are represented in Figure 1, and described in Table 1.



**Figure 1: Computing elements of an IMS and their interactions**

Set	Where set represents
1	BP (ACR's), comprising of API, OS, COEX, Health Monitoring, BITE, and computer module hardware resources with no mapped applications.
2	Applications, comprising of computing code, development environment and analysis tools.
3	Input/Output (IO) devices, comprising of sensors, actuators, display units, error logs, data loaders, and all network hardware.
4	The mapping of applications to BP computing resources, and usage of API by applications.
5	Communications between the applications and IO devices, where BP provides mechanism for passing data between the two.
6	BP interaction with relevant IO devices comprising of error logging, and data loading.
7	Applications outside of IMS with direct access to IO devices, includes device drivers.

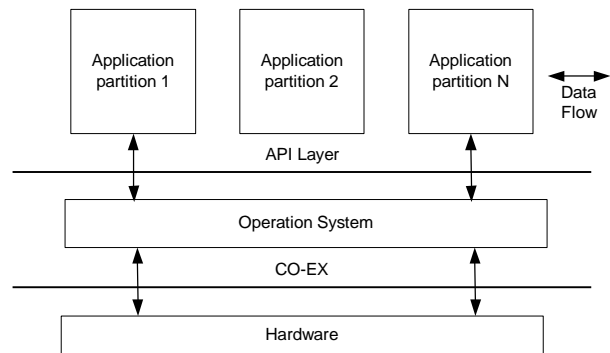
**Table 1: IMS computing elements**

The term IMS BP within this document refers to set 1, the bare computing modules with the layered IMS architecture. The term IMS within this document

refers to those elements in sets 1, 4, 5, and 6. These elements can be seen as the finally configured IMS, ready to run, including all applications and interactions with IO devices.

The next grouping is the set of all IMS computing support elements including those IO devices used for IMS BP support, but without the mapped applications. These are sets 1, and 6. The final group is the application elements mapped onto the modules and the connections with the IO devices used for application support. These are sets 4 and 5. Whilst the modules must provide a robust communications system for set 5, it is appropriate to list this as application dependent as the application ultimately must use or produce the data.

An ARINC 653 (ARINC 1997) variant of an IMS Module model is shown in Figure 2. This model shows several application partitions, and shows that all data flow from the applications should be through an Application Programming Interface (API). All data flow from the operating system (OS) to hardware should go via the Core Executive (CO-EX) layer in order to maintain software portability.



**Figure 2: IMS Module**

If this form of module is employed it becomes, at least for some levels of abstraction, possible to separate out the safety arguments relating to the application (set 2) and the BP (sets 1 and 6). This separation is possible because the failure modes of the BP will not propagate to the application level. In other words, the system must be designed to safely support this separation. This is required to support the arguments for incremental maintenance / certification. We will return to this issue in Section 4.

### A "baseline " safety argument

For the model presented above the skeleton of a safety argument has been produced. This argument would form the basis of the certification of the IMS. It

is called the baseline argument because it provides an index to the safety argument and evidence for the system when it goes into service. The aim is then to maintain this argument basis through the lifetime of the system through an incremental maintenance and certification process.

The top level argument structure for this IMS is presented in Figure 3 in Goal structure Notation (GSN) form (Kelly 1999). This shows the various argument sections required to support the overall goal, “IMS is acceptably safe” A *goal* is a requirements statement expressed as a claim concerning some aspect of the system design, implementation, operation or maintenance. It is represented in GSN by a rectangle. The arrowed lines between goals indicate that the higher level goal is at least partly *solved by* the lower level goal.

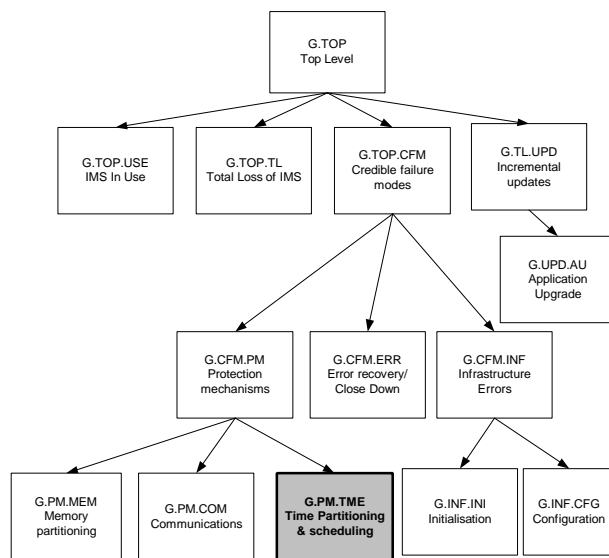


Figure 3: GSN Structure for IMS Model

Consider the safety argument relating to goal G.PM.TME, which considers the response time characteristics of the system. The safety argument presented in this section is not complete as work is ongoing into the structure of the safety arguments for IMS. For instance, it does not cover the issues of timely detection of failures. However, it is sufficient for our current purpose.

This argument is split into two elements: one relating to the timing characteristics of the Bare Platform (IMS services) and one relating to the applications that will be resident on the platform. These structures are presented in Figures 4 and 5 respectively.

Figure 4 indicates that the timing requirements of the BP can be shown to be met if an appropriate set of

timing requirements are provided and it can be shown that these requirements can be met by the implemented system. The circles, such as S.TME.1, represent *solutions*; that is an immediate source of information that can be used to show the relevant goals have been met.

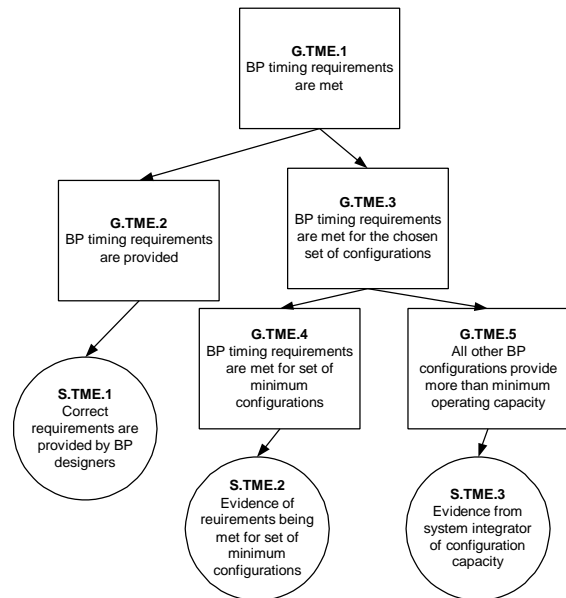


Figure 4: Safety argument for IMS Timing Requirements

Note that the evidence required to substantiate a solution may be provided by different stakeholders in the system. In this case the BP designers must provide evidence of correct timing requirements and the system integrators must provide evidence that BP configurations with more than minimal functionality still meet their timing requirements.

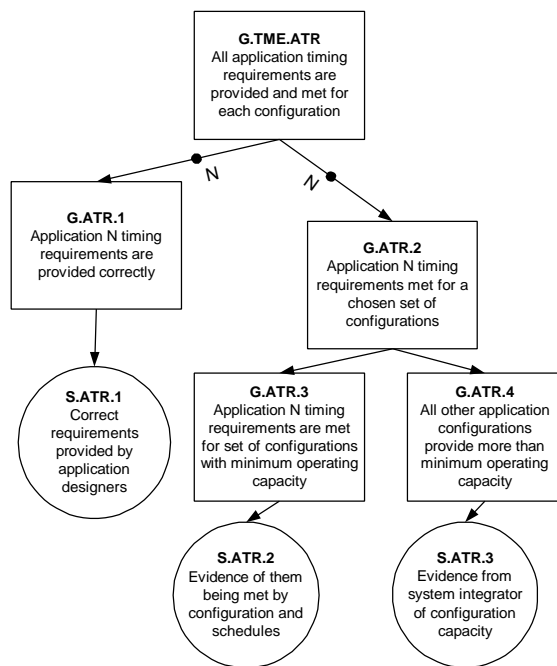
Who provides, and maintains, safety evidence is an important issue in incremental maintenance and certification. This is a contractual and operational process issue that is outside the scope of this paper. However, for incremental maintenance and certification to be successful it must be addressed.

One of the required features of IMS is the ability to provide a look up table of valid system configurations (Nicholson, Hollow et al. 2000). These can be used if a failure is detected in flight to facilitate reconfiguration to a new, and acceptably safe, configuration. Thus, timing requirements must be shown to be met for all chosen configuration sets. The timing requirements issue is partly eased by the time partitioning philosophy employed by the IMS.

The time partitioning philosophy where partitions are scheduled in a cyclic fashion, each with their own internal schedule should provide protection from partitions which are behaving incorrectly. For example

a partition which is babbling or stuck in an infinite loop should not dominate the processor preventing a different partition from meeting its deadlines. However, this philosophy will increase the amount of jitter (Audsley and Wellings 1996) in the schedule at least by an amount equal to time taken for the other partitions to be executed and the modules should be configured carefully with this in mind.

Within the goal structure presented in Figure 5 an arrow with no decoration and a filled arrowhead represents one to one mapping between goals. One to many mappings are also used, and are represented by an arrow with a small circle on the arrow line. This type of mapping indicates that the following goal will need to be met numerous times. In this case it shows that a set of evidence will be required from each application manufacturer.



**Figure 5: Safety argument for Application Timing Requirements**

In this Section we have shown that the GSN approach allows a representation of the safety argument for the IMS to be developed. This representation can then be used to investigate any challenges to the validity of the argument, and the evidence employed to substantiate it, during the operational life of the system. The challenges to the baseline safety argument posed by incremental maintenance of a system are considered in Section 4.

## 4. INCREMENTAL CERTIFICATION

### Types of change

The lifetime of a modern jet is typically 20 to 30 years and during this time the requirements of the avionics system, and the platform technology this functionality will employ, are subject to change. We have identified eight categories of change that our IMS may be subject to:

1. A change to an application, where change effects are contained within a single partition (could include the removal of an application)
2. A change to an application where change effects cross partition boundaries (for instance requiring the movement of an application to a different module)
3. An addition (i.e. involving extra application(s)) where change effects are contained within a partition
4. An addition (i.e. involving extra application(s)) where change effects are not contained within a partition
5. A change / addition of an application that requires the partition boundaries to be altered
6. A change to the hardware platform (involves a change to the Core Executive Interface (coex))
7. A change to the API implementation
8. A change to the operating system

Note that when we discuss a change crossing the partition boundaries, this means that the change does not alter any functional or resource dependency external to the changed partition. These categories can be related to the sets of computing elements presented in Figure 1. For instance change category one may only affect a single element of set 2. Other categories may have an impact on other sets. For example, a change to the API may potentially have an impact on the safety arguments relating to all the sets presented in Figure 1. The ability of a change to impact other elements of the system depends primarily on the strength of the interfaces and partitioning mechanisms (see Figure 2) in the system.

One of the desired advantages of IMS is the ability to provide a process by which incremental maintenance, and incremental certification of the results of that maintenance, can take place. That is the ability to implement some, or all, of these eight categories of change without the need to recertify the entire system. The need to allow for change and growth in the system means that incremental certification is a design as well as a maintenance

issue.

### **Maintenance and design for change**

Two types of maintenance activity can be identified for systems: unplanned and planned. In this context unplanned maintenance relates to changes initiated as necessary during the operational life of the system. Planned maintenance refers to batches of maintenance actions that are undertaken at one time. Currently, mid life updates are undertaken as planned maintenance actions.

In order to deal with unplanned change the system should have built in slack in the size of the memory and time partitions. For instance, a reservation based scheduling approach (Grigg and Audsley 1997) could be adopted. This approach gives each schedulable entity a budget. If it can be shown that if each of the budgets is met then the timing requirements are met. After each change evidence must be provided that each budget is still met. As a result of this approach unplanned changes can be accommodated with minimum impact on the system (i.e. without requiring partition boundaries to be moved). The impact on the safety basis of the system can then be assessed via an incremental certification process.

In some cases the impact of a change which crosses partition boundaries is too wide spread or the slack in the system has been exhausted so cannot accommodate the change. This type of change will have far-reaching effects on the safety case and as such is not easily amenable to incremental certification. These changes are better undertaken a part of a planned maintenance process. The trick is to maximise the number of changes that can be executed with minimum impact, via an incremental certification process. This is the basis of our ongoing research work.

### **Incremental certification process**

The groundwork for incremental certification must be presented in the baseline safety argument. In other words the designers and system integrators must provide an argument showing that suitable provision for incremental certification has been made. Furthermore, a process or roadmap for each category of change of the eight should be identified. These processes must be acceptable to the certification authorities prior to use.

Let us consider a modification to an application, assuming that the underlying IMS platform is not changed. Furthermore, assume this modification will conform to the appropriate development assurance / integrity level and that evidence will be provided by the

application designers that the change conforms to the approved API guidelines.

A process of the kind outlined in Table 2 could be employed. This process considers the elements required to undertake the modification and the impact on the safety arguments / evidence of the modification. This is a process tailored to a change to an application. However, similar approaches can be applied for other modifications, such as an introduction of a new application.

- 1: Confirm that modified/new elements conform to appropriate design assurance / integrity level and API usage criteria
- 2: Analyse existing system to identify the impact of the proposed modification.
- 3: Identify the certification requirements for this change based on an appropriate certification roadmap.
- 4: Instantiate the roadmap to provide arguments / evidence for the certification authorities. It is likely that all of the roadmaps will:
  - 4a: Provide analysis of potentially affected applications for any safety implications. Check that assumptions are not invalidated. This should include an analysis of initialisation, fault tolerance and operational running of the system
  - 4b: Provide a qualified data loading process to ensure actual upgrade does not introduce any hazardous conditions.
  - 4c: Provide a configuration management process to ensure that the master configuration list is updated with the modified configuration.
  - 4d: Provide evidence of suitable testing to ensure the unaffected applications really have been unaffected and any incorrect operation of the affected applications is detected on initialisation.
  - 4e: Provide evidence to show the impact on overall flying characteristics of the aircraft of the modification. Show that any changes in understanding by the aircrew of how the systems operate in both normal and failure conditions have been communicated to the crew.

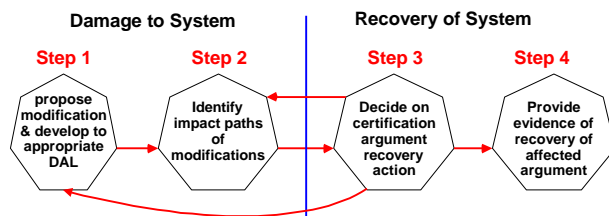
### **Table 2: Incremental Certification Process**

One aim of the process is to try to minimise the impact of a change on the safety basis for the aircraft and hence maximise the number of changes that can be accommodated via incremental certification. The amount of work to be undertaken for a change can be minimised by using the concept of equivalence. Under

equivalence we certify a “family” of variants of a configured IMS by showing that they have properties no worse than a representative member of the family.

If we can show that the impact of the modification to the application on other elements of the system is such that the safety argument for the new system is equivalent to the old argument, with minimum effort, then we can provide incremental certification. Thus, this equivalence family represents the set of changes that can be made in accordance with the original safety basis.

The steps required to provide an equivalent argument to the original safety basis for the system provide a certification roadmap for the system, mimic the steps introduced in Table 2 and are shown in Figure 6. Again, a different roadmap may need to be provided for each of the categories of change that can be accommodated with the incremental certification approach.



**Figure 6: Maintaining the Safety Basis**

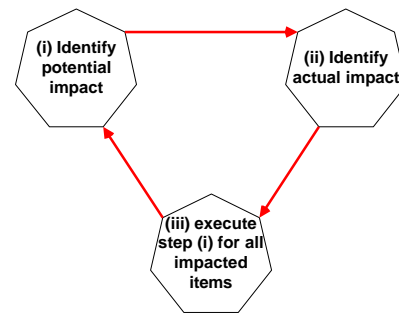
To produce an equivalent safety argument as part of incremental certification we must first of all consider the impact paths of the modification. Two types of impact on the safety argument can be envisaged: those that directly challenge the evidence, or goals, provided in the baseline argument and those that indirectly challenge the baseline via their impact on the underlying assumptions within the baseline argument.

In Figure 7 (Kelly 1999) a generic impact analysis is presented. This analysis can be tailored to consider the direct impact of a change on the safety arguments and safety evidence provided in the baseline safety case.

The three elements are:

- (i) identify the potentially impacted elements and relations according to *rules* that address the set of possible impacts
- (ii) from the potential impact identify the actual impacted elements and relations - at the same time determining which of the impacts can be considered to be *benign*

- (iii) repeat process by executing step (i) for all identified (actual) impacted items



**Figure 7: Impact Analysis**

Step (iii) is a recursive call that allows knock-on effects of a modification to be tracked. The process stops when either the impacts of a modification cannot be propagated any further or all remaining impacts are considered to be benign.

One aim of IMS is to reduce the propagation of impacts as much as possible. This represents a key difference between IMS and federated systems. IMS polices the segregation between applications and provides hardware abstraction, bounding the propagation of the impact of a change. Federation on the other hand is implemented in a way that means that re-analysis and recertification of the whole system is often the only viable approach.

The impact analysis can also be used to investigate the impact of the modification on the function, timing and data attributes of the system. The impact paths from this analysis provide the indirect challenges to the safety argument. For instance, a challenge to an assumption that the probability of a failure of the IMS infrastructure is less than  $10^{-9}$ . It may also be the case that combinations of failures introduced by the modification are worse than the existing individual failure. Using the impact analysis, along with the boundaries that the effects cross (application, partition, API, OS, platform) will help determine the ability to produce an equivalent safety argument.

Once the impact of the modification has been addressed and the potential impacts on the baseline safety case have been determined recovery actions are required to produce an equivalent safety case. Each of the different categories of change introduced above will require different actions to produce an equivalent safety case. Each set of actions forms the incremental certification roadmap for that form of change.

Consider the introduction of a modification to an application where the impact of change does not cross partition boundaries. In this case the strategy to

separate the application arguments from each other and from the IMS service arguments limits the impact on the safety argument. Thus, arguments relating to the IMS structures (sets 1, and 3 to 7) will not be directly challenged by the change.

Once all the challenges to the baseline safety case have been addressed the modification can go ahead. This will involve the actions put forward in step 4 in Table 2. As a result of this activity the modification can take place and the safety basis of the system can be maintained.

### Discussion

Incremental modification, incremental certification and the processes required to undertake them are new and novel features of an IMS approach. The approach presented above represents the first steps towards an analytical framework for these activities. It is therefore worth introducing an example and looking at the areas where effort must be placed if the aims of incremental certification are to be achieved.

Let us consider the potential impact of a change on the safety argument for the timing attributes of a system. In GSN terms a potential challenge to an element of the argument is represented by a cross through the challenged link.

The baseline argument for this attribute was presented in Figures 4 and 5. Goal G.TME.1 is not challenged by a change to an application that conforms to the API guidelines, see Figure 8. This shows the advantage of structuring the safety argument so that arguments about the IMS structure are not affected by changes to the applications that use the facilities.

The goal G.TME.ATR is potentially challenged by the change, see Figure 8. For instance, the evidence that appropriate timing requirements for the application have been provided is challenged. Furthermore, the evidence that the requirements are met has also been challenged.

Note that these arguments aid incremental certification. For instance, there are  $N$  goals relating to the meeting of timing requirements (G.ATR.3). The argument is structured in such a way that those applications whose response time characteristics are not impacted by the change need not be re-evaluated. Even those that are re-evaluated if they continue to meet their deadlines their sections of the argument are not impacted.

Consider, the problem of handling applications in a distributed system where it is required for a message generated on one processor to carry useful information to a function on another processor. The latter function

in this application should not be released until the message has arrived. One approach was suggested by Liu and Sun (Sun and Liu 1996), and was referred to as the Phase Modification Approach.

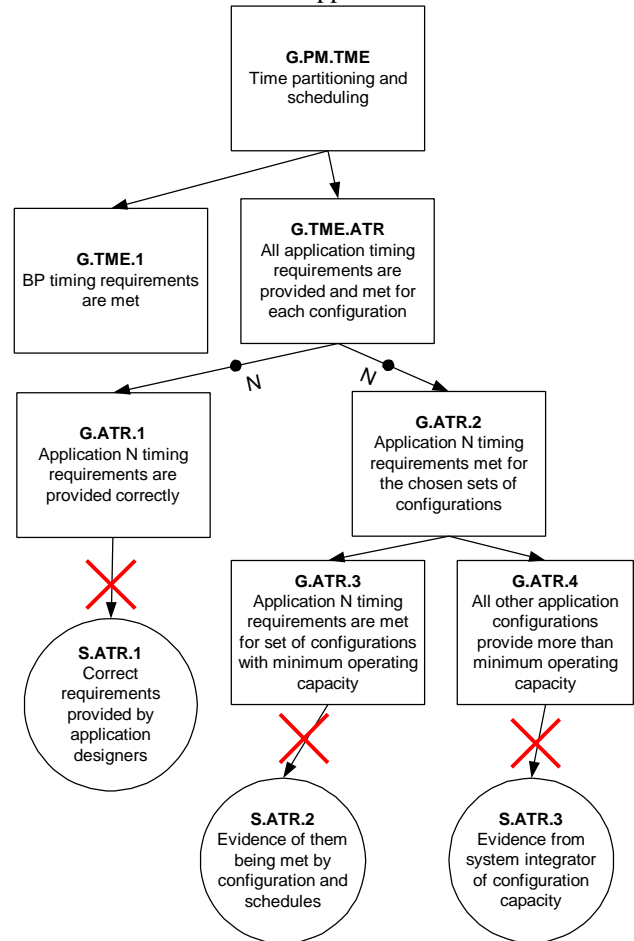


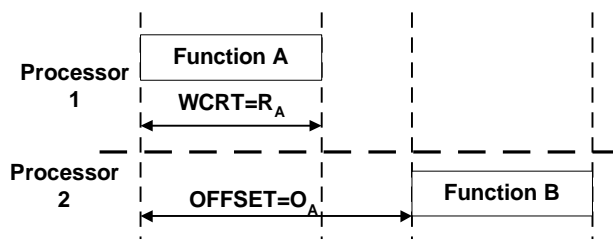
Figure 8: Challenges to the Application Timing GSN

By giving a function an offset such that its dispatch (or release for simplicity) is always greater than the worst case response time of the event trigger (i.e. the worst case arrival time of the message) precedence is maintained, even across a distributed system. This approach would require a global time base.

Another way of handling distributed transactions is to use sporadic functions that are triggered when the message has arrived, however this suffers from the same problem that the approach by Sun and Liu had. That is, a change on one processor means that the timing characteristics of the whole system have to be re-analysed. Therefore, these approaches are inappropriate for our needs.

In (Bate 1998), an alternative approach was proposed to help solve these problems and to reduce pessimism. The difference between our approach and that of Liu and Sun is that instead of making the offset equal to the worst case response time of the event trigger, a larger value is selected based on a number of timing-related criteria. One of the benefits of doing this is the amount of re-verification is reduced when the system changes. The reason is that within defined bounds a change on one processor does not necessitate a system wide re-verification, i.e. it is not holistic.

Figure 9 illustrates how the timing characteristics of processor 1 may be modified until  $R_A \geq O_B$ , without affecting the scheduling of processor 2, since  $D_A = O_B$ . Therefore, if the software on one processor is modified, then only that processor needs to be re-analysed as long as the overall system is schedulable. When the timing requirements are no longer met, the timing analysis and function attribute assignment for the whole system is repeated. In Sun and Liu's approach  $R_A = O_B$  and therefore a reanalysis is required each time a change is made.



**Figure 9: Non-holistic scheduling**

Once any challenges to the safety basis of the IMS have been identified steps 3 and 4 of the process outlined in Figure 6 can be implemented. That is a recovery action decided on and evidence that it has been successful provided. The aim here is to make this action as small as possible in all cases.

The action required to repair the requirements goal (G.ATR.1) is straight forward: provide evidence for the requirements. The repair action to show that application timing requirements have been met, goal G.ATR.3, is dependent on the impact of the change. If sufficient leeway has been provided under the design for growth strategy then the only application that will need to be reanalysed will be the application being changed (for timing). This is the most desirable circumstance.

As the disruption to other applications increases more analysis is required. If, for instance, the time partitioning boundaries need to be changed, e.g. to accommodate a greater worst case execution time, then considerable re-analysis effort may be required to fix

the safety argument presented by G.ATR.2. Thus the further down the list of changes presented above the more effort will be required to show that the new argument is equivalent to the old one.

The challenges to the argument for goal G.TME.ATR do not require a structural change to the safety argument to be resolved. It is therefore fairly easy to show that the new argument is equivalent to the old one. In general, it will be much more difficult to show equivalence if the structure of the argument must change as well. For instance, if any of the assumptions on which the strategies employed in the goal structure for the system are based are challenged, more extensive changes to the safety argument for system will result. An example might be tighter functional integration meaning application evidence would be much harder to separate. Equivalence is then either lost or more difficult to re-establish.

The aim would be to restrict the impact of a change to as few levels as possible as this would restrict the rework required of the safety argument. The levels are individual function, application, system and system of systems. Clearly any timing change that impacts at the system of system level will have a serious impact on the safety argument and evidence.

The aim of any approach to incremental certification therefore must first be to structure the attributes of the IMS and its accompanying safety argument in such a way that the challenges to these structures are minimised for a feasible system change. The explicit use of an OS and API should make this much easier for changes to an application. This puts an onus on the system designers to devise interfaces that support the safety argument. Second, a way of indicating the impact of the different types of change should be devised. This would also indicate how much effort is likely to be needed to maintain the certification basis. We maintain that an incremental certification roadmap for each level of change will facilitate this process.

## CONCLUSIONS

Producing and maintaining a safety case for IMS is non trivial. We have highlighted some of the difficulties and possible approaches to the production of an appropriate argument and accompanying evidence that could form the safety basis for the certification of an IMS. Furthermore, we have shown that the structure of this argument, such as splitting the core IMS and application arguments can facilitate the objective of incremental certification for a subset of the possible changes to an IMS. In particular, we have

shown that incremental certification can only be achieved if an integrated design and safety maintenance process is developed. That is the design must support the incremental certification process.

However, if the promise of incremental maintenance and certification is to be realised a number of issues remain to be resolved. These include accepted road-maps for each type of change to the system, which changes can be dealt with an incremental process and which need to be undertaken as part of a planned maintenance activity, detailed analysis of activities to be undertaken during the change process, etc. The synthesis of the safety basis and incremental certification is the focus of our future research activities. We aim to show that this synthesis will benefit both the designers and maintainers of integrated modular systems.

## REFERENCES

ARINC (1997). Avionics Application Software Standard Interface, ARINC.

ARINC (1999). Design Guidance for Integrated Modular Avionics, ARINC.

Audsley, N. and A. J. Wellings (1996). Analysing APEX Applications. Proceedings of Real-Time Systems Symposium.

Bate, I. (1998). Scheduling and Timing Analysis for Safety-Critical Systems. Department of Computer Science. York, University of York.

Bradley, H. J., F. J., et al. (1996). Integrated Modular Avionics & Certification - An IMA Design Team's View. *IEE Seminar: Certification of Ground/Air System*, Savoy Place, London, WC2R 0BL.

Burns, A. (1995). "A Preemptive Priority-Based Scheduling: An Appropriate Engineering Approach." Advances in Real-Time Systems 239.

DoD (1996). Mil-std 882C.

Grigg, A. and N. C. Audsley (1997). Towards the Timing Analysis of Integrated Modular Avionics Systems. ERA Avionics Conference and Exhibition, ERA.

Kelly, T. (1999). Arguing Safety: A systematic Approach to Managing Safety Cases. Computer Science. York, University of York.

MoD (1996). Safety Management requirements for defence Systems, MoD.

MoD (1997). Requirements for Safety Related Software in Defence Equipment, MoD.

Nicholson, M., P. Hollow, et al. (2000). Approaches to Certification of Reconfigurable IMA Systems. INCOSE, Minneapolis, USA, INCOSE.

SAE (1996). Aerospace Recommended Practice ARP 4754: Certification considerations for highly

complex aircraft systems.

Sun, J. and J. W. S. Liu (1996). Synchronization protocols in distributed real-time systems. 16th International Conference on Distributed Computing Systems.