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 valuable capabilities including 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 investigates the different types of upgrade that an IMS may be subject too. A method is proposed for determining the impact of a proposed upgrade. A baseline safety case for IMS in which evidence can be separated between different stakeholders in the system is presented. This separation facilitates incremental certification by allowing the impact of a change on the baseline safety case to be minimised.

1. Introduction

The current generation computer based systems in commercial jet aircraft 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 are expensive to develop and certify. As a result there has been a move towards an alternative approach known as *Integrated Modular Systems (IMS).* A number of current military aircraft, such as the F-18, have a mixture of federated and IMS like systems. The aim of this paper is to show how to approach certification for aircraft computer based systems that employ IMS.

Federated 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 (safety case) 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 new, rather than reworked, safety case is often produced. Thus, much of the previous information is either lost, or work undertaken that is not required.

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

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 (incremental certification), 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 with consequent reduction in cost of ownership.

The use of IMS changes traditional system architectures. 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 IMS has on the *safety certification* of the system and the maintenance of that certification through the lifetime of a platform. Then Section 2 presents background information on IMS, certification of an IMS, and the issues arising from modifications to such systems. In Section 3 we propose a process for handling the impact of a modification on the safety certification basis of an IMS.

Section 4 investigates the use of Goal Structuring Notation (GSN) as a basis for developing and maintaining the safety arguments on which the safety certification of a system is based. The GSN representation facilitates the process presented in Section 3. Emphasis is placed in this section on the ability to separate the safety arguments relating to the core services provided by the IMS, and the applications that reside on the IMS, for initial certification. The concept of a set of "logically equivalent" systems (Nicholson, Hollow et al. 2000), and system safety arguments, is then used as a unifying theme for maintaining the safety arguments, and hence safety certification, in the face of incremental modifications to the system.

Finally, in Section 5 a case study is presented. This study looks at the impact of a change on the timing characteristics of an IMS.

2. Background and Problem Statement

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 brainstorming list was "*What is IMS*"? Therefore, we have chosen one particular exemplar that has the advantage of being based on a standard, ARINC 653 (ARINC 1999).

An ARINC 653 IMS contains a number of computing modules that are grouped within cabinets that are positioned throughout the platform. Modules and cabinets 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. We will return to this issue in section5.

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, see Figure 1 and Table 1.

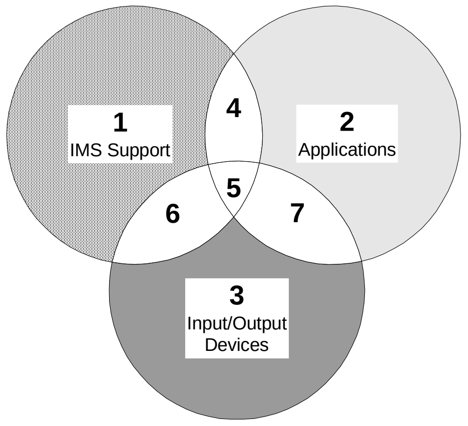


Figure 1: Computing elements of an IMS and their interactions

Set	*Elements*
1	BP, comprising of API, OS, CO/EX, 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 indicates 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.

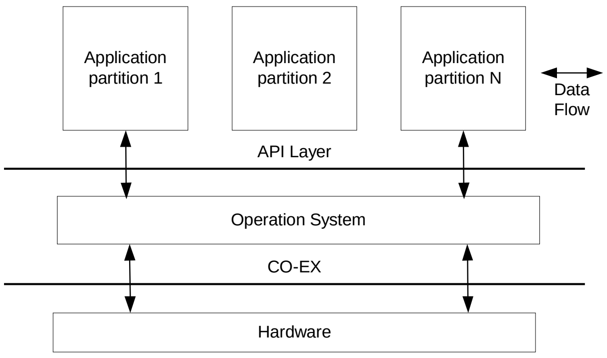


Figure 2: IMS Module

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 applications (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 also supports the arguments for incremental maintenance/ certification. We will return to this issue in Section 4.

Certification of Avionics Systems

Under current civil aviation certification procedures each computer-based system on an aircraft is assessed in isolation and the certification process assumes that the software behaves in a deterministic manner. Recent international standards such as ARP-4754 / 4761 (SAE 1996) 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. For example, modifications are considered in section 11 of ARP 4754. However, this section does not explicitly address the implications of incremental modification, including the need to maintain the safety argument through the lifetime of the IMS. Thus, the standards do not help when considering this aspect of IMS. There is a need to go back to more basic principles. This boils down to providing a safety argument and maintaining it throughout the lifetime of the system via a process of incremental certification.

Incremental Modification and Certification

Incremental certification provides the ability to integrate and qualify new applications, and maintain existing applications, without the need to re-qualify the whole platform. The principle of incremental modification has been used on previous projects, such as the F-18. However, the impact on the safety arguments resulting from these changes is not well understood.

A method of determining the set of changes that can be undertaken on an incremental basis, and the impact on the safety case of such changes, is therefore required. In the remainder of this section the types of changes that an IMS will be subjected to are presented.

The lifetime of a modern jet is typically 20 to 30 years and during this time the requirements of the computer based systems, and the platform technology this functionality will employ, are subject to 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, such as the replacement of a failed processor. 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 some cases the impact of a change may be extensive and have far-reaching effects on the safety case and as such will not be amenable to incremental certification. These changes are better undertaken as part of a planned maintenance and re-certification 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 the approach presented in Sections 3 and 4.

Consideration of the characteristics of the ARINC 653 IMS has led to eight categories of change being identified:

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 if a change does not cross a partition boundary, this means that the change does not alter any functional or resource dependency external to the changed partition. Some of these changes will need to be undertaken as planned activities, that require re-certification, and others as unplanned activities, that can be accomplished via incremental certification.

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 the IMS set (1,4,5 and 6) presented in Figure 1.

The ability of a change to impact other functional elements of the system depends on the nature of the functional dependencies in the system and the API and partitioning mechanisms (see Figure 2) in the system. For example, if the safety argument for an application relies on it having access to a particular piece of data, that as a result of a modification is also used by another application, incremental certification can be achieved if the modified application cannot significantly affect the ability of the existing application to access that data.

In the next section means of identifying the impact of a modification on an IMS is presented. Particular emphasis is given to the ability to identify the impact of a proposed modification on the safety arguments that form the basis of the safety certification for an IMS. Some of the eight change types identified above will have minimal impact on the safety arguments and therefore will be amenable to incremental certification. Others will have a wider impact and will require re-certification via a planned maintenance activity. A key element of IMS design is to minimise the number of changes which fall into this class.

3. Process for Handling Change

The impact of a modification is dependent on a number of factors. One of these is the amount of "slack" in the system. It is clear that for an incremental update to be viable for IMS a system should be built with some slack in the size of the memory and time partitions. This is known as the design for growth strategy. For instance, a reservation based scheduling approach (Grigg and Audsley 1997) could be adopted. This approach gives each schedulable entity a budget. It can be shown that if each of the budgets is met then the timing requirements for the system are met. For each modification, evidence must be provided that each budget is still met. As a result of this approach modifications 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.

The groundwork for incremental certification is presented in the baseline safety argument. In other words the designers and system integrators provide an argument showing that suitable provision for incremental certification has been made. Furthermore, a process or roadmap for each of the eight categories of change identified in Section 3 is required. The exact nature of each roadmap is not obvious. These processes must be acceptable to the certification authorities prior to use. Note that in some cases the roadmap will imply full re-certification rather than incremental certification. Work remains on developing the details of the roadmaps.

To maximise the number of types of modification that can be addressed via incremental certification the impact of a change on the safety basis for the aircraft should be minimised. This is facilitated by employing the concept of "logical equivalence". Under logical equivalence a “family” of variants of an IMS is certified by showing that each member of the family has acceptably similar properties. For in a family instance, response times of the applications meet their timing requirements and hence the timing strand of the safety argument is not broken, although the evidence required to support it will be different, see Section 5.

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

Consider a modification that is likely to be amenable to incremental certification: a modification to an application. A process of the kind outlined in Table 2 could be employed. This process considers the steps required to undertake the modification and the impact on the safety arguments/evidence of the application modification. Similar approaches can be undertaken for other modifications, such as the introduction of a new application.

The steps in Table 2 are required to show whether the modified system is logically equivalent to the old system and therefore can be certified under the original safety basis. Note that this does not imply that there will be no impact on the arguments and evidence in the safety case, only that the new safety case is logically equivalent to the old one.

In step 2 the impacts of the proposed modification are assessed. Two types of impact on the safety argument can be envisaged: those that directly challenge the arguments or evidence provided in the baseline argument and those that indirectly challenge the baseline via their impact on the underlying assumptions within the baseline argument.

1:	Confirm that modified/new elements conform to appropriate design assurance level (design rules) and API usage criteria for an ARINC653 application
2:	Analyse existing system to identify the impact of the proposed modification. Functional, non-functional (e.g. timing) and safety argument impacts should be investigated.
3:	Identify the certification requirements for this change. That is the certification roadmap for an application modification in an ARINC 653 system
4:	Instantiate the roadmap to provide arguments / evidence for the certification authorities. The roadmap will:
4a:	Provide analysis of any potential safety implications. For instance, 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 any impacts on the flying characteristics of the aircraft implied by a given modification. Show that any changes in understanding required by the aircrew of how the systems operate in both normal and failure conditions have been addressed.

Table 2: Incremental Certification Process

In Figure 3 (Kelly 1999) a generic impact analysis is presented. This analysis can be tailored to consider the direct impact of a modification on the safety arguments and safety evidence provided in the baseline safety case, in support of step 2 above.

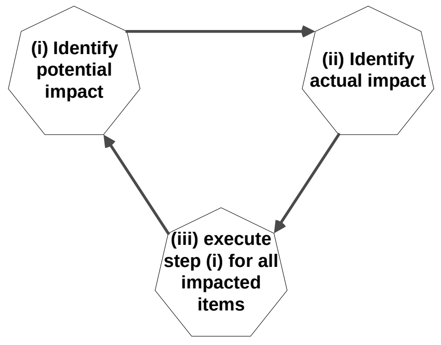

(i) Identify potential impact
(ii) Identify actual impact
(iii) execute step (i) for all impacted items

Figure 3: Impact Analysis

Step (i) investigates the "damage" to a system that may result from a modification. Analysis starts at the intra-partition effects, then the inter-partition, inter module and inter cabinet effects via the recursive call in step (iii).

Step (ii) considers the scope of the impacts in terms of the safety case for an IMS. A *benign* impact is one where logical equivalence can be shown between the existing and new safety arguments. That is the safety evidence implies the safety characteristics of the modified system are acceptably similar to those of the original system.

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 on the safety arguments / evidence, resulting from a modification, 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 the dependencies between applications, the data they use and the non-functional characteristics of the system (e.g. timing) are not easily identified. As a result re-analysis, reconstruction of a significant part of the safety arguments, and evidence, and hence recertification is often undertaken.

Impact analysis can also be used to investigate the impact of the modification on the function, timing and data attributes of the IMS. 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 any existing failures. Using impact analysis, along with the boundaries that the effects cross (application, partition, API, OS, platform) will help determine the ability to produce a logically 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. Step 3 of the process in Table 2 introduces the concept of a certification roadmap for a modification. In Figure 4 the roadmap for a change is presented in terms of the recovery actions required to maintain the safety arguments for the system being modified. This set of actions forms the basis of step 4 in the process. A different roadmap is needed for each of the categories of change.

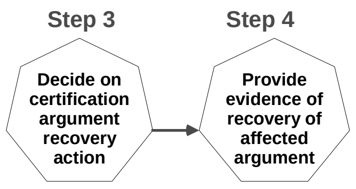

Step 3
Step 4
Decide on certification argument recovery action
Provide evidence of recovery of affected argument

Figure 4: Maintaining the Safety Basis

Consider the introduction of a modification to an application that does not cross partition boundaries. In this case the appropriate strategy is to separate the application arguments from each other and from the IMS service arguments. Thus, arguments relating to the IMS structures (sets 1 and 3 to 7) will not be directly challenged by the change. In Section 4 we contend that separating application and IMS structure safety arguments is an important step towards incremental certification of an IMS.

Once all the challenges to the baseline safety case have been addressed the modification can go ahead. As a result of this activity the safety basis of the system can be maintained.

Incremental modification, incremental certification and the processes required to undertake them are central to the gains that can be made from adopting an IMS approach to constructing computer based systems on aircraft. The framework presented above provides the basis of this work. It is clear that there is a need to be able to quickly identify the potential impacts (challenges) to the safety basis posed by a modification. This is by no means a trivial requirement. However, techniques exist to present safety arguments. In the next section we consider how the incremental certification process can be supported by the use of Goal Structuring Notation.

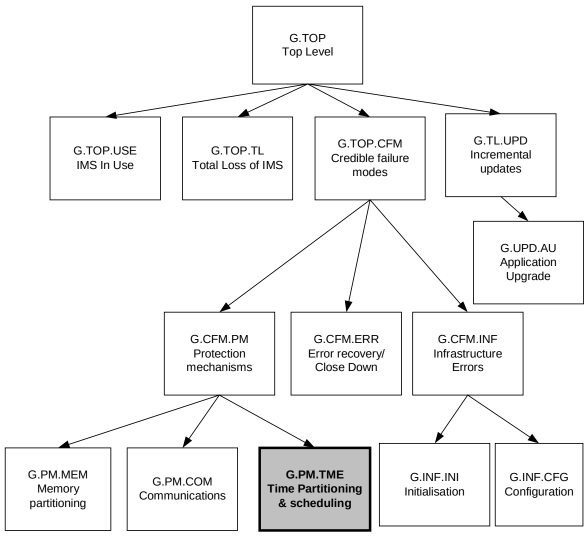
4. Safety arguments to Support Change

The Goal Structuring notation (GSN) (Kelly 1999) can be used to present the arguments for the initial certification of the platform. It can also be used to investigate the impact of a modification on this safety argument, step 2 of the process introduced above. Let us first consider the production of a GSN representation of the "baseline" safety argument for an IMS.

A "baseline " safety argument

For the ARINC IMS model presented in Section 2 the skeleton of a safety argument has been produced. 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 to maintain this argument basis through the lifetime of the system through an incremental maintenance process.

The top level GSN argument structure for an ARINC 653 IMS is presented in Figure 5. This shows the various argument sections required to support the overall goal (G.TOP), "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.



G.TOP Top Level
G.TOP.USE IMS In Use
G.TOP.TL Total Loss of IMS
G.TOP.CFM Credible failure modes
G.TL.UPD Incremental updates
G.UPD.AU Application Upgrade
G.CFM.PM Protection mechanisms
G.CFM.ERR Error recovery/ Close Down
G.CFM.INF Infrastructure Errors
G.PM.MEM Memory partitioning
G.PM.COM Communications
G.PM.TME Time Partitioning & scheduling
G.INF.INI Initialisation
G.INF.CFG Configuration

Figure 5: GSN Structure for IMS Model

Note that the structure in Figure 5 contains a goal for incremental update. This goal is met by presenting an argument for the maintenance of the safety basis, via a certification roadmap, for each of the eight types of modification. The application modification goal is shown as an example.

Developing the safety arguments in such a way that the impact of a modification is minimised supports incremental certification. One clear division in an IMS is between the bare platform and the applications. If the safety argument for the platform is developed so that the arguments relating to these two elements are separated then a substantial reduction in the amount of reworking of the certification basis can be achieved. It might for example be that some modifications to the bare platform, such as a bug fix to the OS, may be transparent to the applications. As a result of the separation of the safety arguments there would be no need to revisit the strands of the baseline safety argument relating to the applications.

In Figure 5 the goal G.PM.TME relates to the safety of the timing characteristics of the system. This goal may be broken into two sub-goals relating to the timing requirements of the BP (G.TME.1) and the timing requirements of the applications (G.TME.ATR). In Section 5 we show how this separation facilitates impact analysis and incremental certification of an application modification.

The GSN representation of the safety argument for an IMS can be used to investigate any challenges to the validity of a safety argument, and the evidence employed to substantiate it. In GSN terms a cross through the challenged link represents a potential challenge to an element of an argument. Suppose that a modification challenges the basis of the argument for a safe initialisation process (G.INF.INI). In this case the arrow between G.CFM.INF and G.INF.INI will be "crossed out". The impact of a modification can be propagated up the goal structure. In this case propagation will only occur if G.INF.CFG is also challenged. This propagation is a specialisation of step 2 in our process.

Once any challenges to the safety basis of the IMS have been identified steps 3 and 4 of the process outlined in Figure 4 can be implemented. That is a recovery action decided on and evidence that it has been successful provided. For instance, one action may be to change the initialisation procedures to overcome the challenges identified above. This action forms part of the roadmap in step 4 of the process. Note that the change in procedures may be extensive enough to require recertification. In this case logical equivalence cannot be maintained in an incremental manner. However, deciding whether this is the case is we believe easier using our approach than is currently the case.

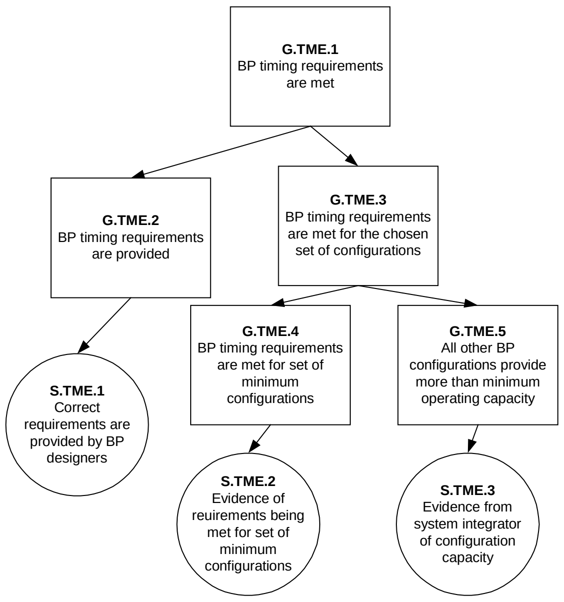
The aim of any approach to incremental certification therefore must first be to structure the attributes of the IMS and its accompanying safety argument in such a way that the challenges to these structures are minimised for a feasible system change. The explicit use of an OS and API should make this 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 in an incremental manner. We maintain that an incremental certification roadmap for each type of change will facilitate this process.

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.

5. Case Study - Changes within the Timing Domain

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, i.e. to illustrate the approach.

The argument relating to G.PM.TME can be 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 6 and 7 respectively. These argument structures have been simplified for purposes of explanation.



G.TME.1 BP timing requirements are met
G.TME.2 BP timing requirements are provided
G.TME.3 BP timing requirements are met for the chosen set of configurations
G.TME.4 BP timing requirements are met for set of minimum configurations
G.TME.5 All other BP configurations provide more than minimum operating capacity
S.TME.1 Correct requirements are provided by BP designers
S.TME.2 Evidence of reuirements being met for set of minimum configurations
S.TME.3 Evidence from system integrator of configuration capacity

Figure 6: Safety argument for IMS Timing

Figure 6 indicates that the BP timing requirements have been met if an appropriate set of timing requirements are provided and 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.

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 ARINC IMS.

The time partitioning philosophy where partitions are scheduled in a cyclic fashion, each with their own internal schedule should provide some protection from partitions which are behaving incorrectly. For example a partition which is 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 the time taken for the other partitions to be executed and the modules have to be configured carefully with this in mind.

Figure 7 indicates that the timing requirements of the applications 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. Within the goal structure presented in Figure 7 an arrow with no decoration and a filled arrowhead represents one to one mapping between goals. One too 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.

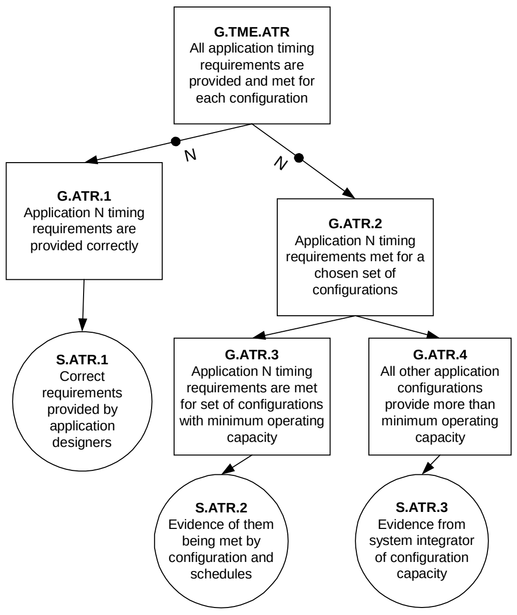

G.TME.ATR
All application timing requirements are provided and met for each configuration
N
N
G.ATR.1
Application N timing requirements are provided correctly
G.ATR.2
Application N timing requirements met for a chosen set of configurations
S.ATR.1
Correct requirements provided by application designers
G.ATR.3
Application N timing requirements are met for set of configurations with minimum operating capacity
G.ATR.4
All other application configurations provide more than minimum operating capacity
S.ATR.2
Evidence of them being met by configuration and schedules
S.ATR.3
Evidence from system integrator of configuration capacity

Figure 7: Safety argument for Application Timing Requirements

Consider the potential impact of a modification to an application on the safety arguments presented in Figures 6 and 7. 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.

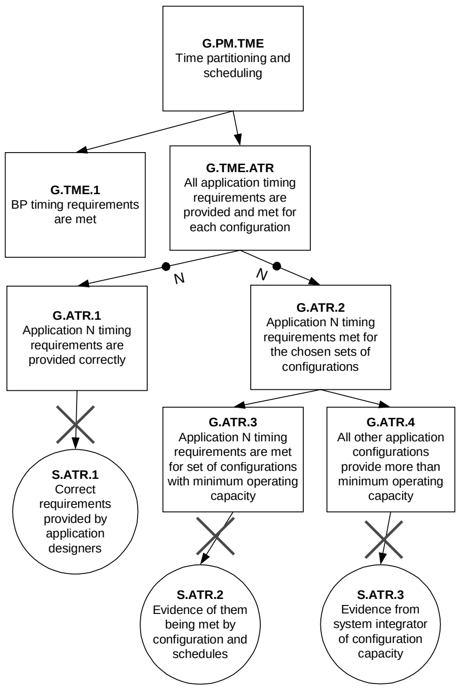

G.PM.TME
Time partitioning and scheduling
G.TME.1
BP timing requirements are met
G.TME.ATR
All application timing requirements are provided and met for each configuration
N
N
G.ATR.1
Application N timing requirements are provided correctly
G.ATR.2
Application N timing requirements met for the chosen sets of configurations
G.ATR.3
Application N timing requirements are met for set of configurations with minimum operating capacity
G.ATR.4
All other application configurations provide more than minimum operating capacity
S.ATR.1
Correct requirements provided by application designers
S.ATR.2
Evidence of them being met by configuration and schedules
S.ATR.3
Evidence from system integrator of configuration capacity

Figure 8: Challenges to Application Timing

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. As a result of the challenge to S.ATR.3 and S.ATR.4 the goal G.ATR.2 is also challenged.

Note that this 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 impacted if they continue to meet their deadlines are said to be subject to a benign impact.

Approaches to Scheduling to aid Incremental Certification

The scheduling approach employed in an IMS can reduce the impact of a modification on the timing characteristics of the applications, and hence the system. 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.

By giving a function an offset such that its dispatch (release) 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. A similar analysis can be undertaken to investigate best case response times. This approach requires 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 as Sun and Liu's approach. 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. This approach makes the offset greater than the worst case response time of the event trigger, based on a number of timing-related criteria. One of the benefits of doing this is that re-verification effort 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. As a result showing that the new schedule is equivalent to the old one is relatively easy.

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. At this point G.TME.ATR is challenged, see Figure 8. In Sun and Liu's approach $R_A = O_B$ and reanalysis is required each time a change is made.

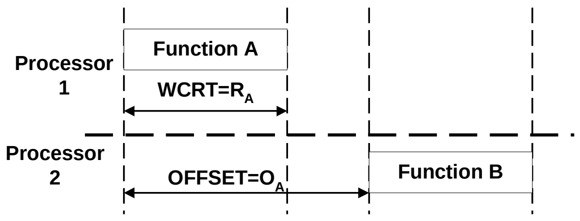


Figure 9: Non-holistic scheduling

The action required to repair the requirements goal (G.ATR.1) is to provide evidence for the requirements implied by the modification. 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 Bate's analysis can be used to produce a new schedule that can be shown to be logically equivalent to the old schedule.

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 worse case execution time, then considerable re-analysis effort may be required to fix the safety argument presented by G.ATR.2. A change to the API, for instance, may mean that the timing properties of all applications must be reassessed. Thus the further down the list of changes the more effort is required to show logical equivalence.

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 possible to show that the new argument is logically equivalent to the old one.

6. 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 the design supports 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 roadmaps for each type of change to the system, which changes can be dealt with by an incremental process and which need re-certification, detailed analysis of activities to be undertaken during the change process, etc. The synthesis of the certification 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.