

## Safety Analysis and Certification of Open Distributed Systems

P. M. Conmy; Department of Computer Science, University of York, York, YO10 5DD U.K.

M. Nicholson; Department of Computer Science, University of York, York, YO10 5DD U.K.

Y. Purwantoro; Department of Computer Science, University of York, York, YO10 5DD U.K.

J. A. McDermid; Department of Computer Science, University of York, York, YO10 5DD U.K.

Keywords: Integrated Modular Avionics, Safety, Certification, Maintainability

### Abstract

An open distributed system is a network of computer modules upon which one or more computer applications can run and inter-communicate. Recently there has been a move to use an open distributed system, known as Integrated Modular Avionics (IMA), aboard aircraft. To fully utilise the benefits of IMA, such as incremental upgrade, a change is needed to the safety analysis and certification of avionics computer systems. At present aircraft computing systems are, generally, physically separate units, with software tightly bound to the underlying computing hardware. Individual systems analysis examines the unit as a whole entity, producing a monolithic safety case. Thus, even a small change in the system can mean that extensive re-analysis is required. Clearly if incremental upgrades are to be achieved for a system as complex as IMA a different approach is required. This paper presents a framework for certification which adapts existing ARP standards, detailing a number of “consolidating interaction” activities that are required between the operating system and applications developers and safety analysts during the development life-cycle in order to facilitate largely independent safety analysis of the components, whilst still ensuring that the system is acceptably safe. We demonstrate how these activities will help establish a maintainable safety case for IMA.

### Introduction

An open distributed system is a network of computer modules upon which one or more computer applications can run and inter-communicate. The applications interact via the network, using a well-defined interface. These systems provide potential benefits such as reconfiguration in the presence of failure and incremental change of software and hardware with minimal impact on other components in the system. Recently there has been a move to use an open distributed system, known as Integrated Modular Avionics (IMA), aboard aircraft.

To fully realise the benefits of IMA a change is required to the way avionics computer systems are analysed and certified. At present analysis techniques are based on the assumption that systems are federated, that is, each individual computer application runs on its own dedicated hardware unit. Within the unit, software is tightly coupled to the hardware. Analysis of individual systems examines each unit as a whole, resulting in a monolithic safety case. Thus, even a small change in the system can mean that extensive re-analysis is required. At the aircraft level, failures of individual systems can be assumed to be largely independent due to the physical separation of the units (although there may be some unavoidable dependencies).

Certification and analysis of an IMA system can be undertaken using current techniques. However, any change in the system, including software to hardware mapping, would require extensive re-analysis and would be prohibitively expensive. Further, it would negate the reasons for introducing IMA. Therefore, a different approach to certification and analysis is required which is based on logical rather than physical boundaries. Components such as the operating system, applications, and computing hardware need to be analysed independently, with their interactions being examined at the interface level. Other aspects of IMA design, such as memory protection and scheduling, also need to be examined to demonstrate a level of independence between components.

This paper presents an approach to certification and analysis of IMA systems based on the ARP 4754 (ref. 1) and ARP 4761 (ref. 2) standards. We detail a number of “consolidating interaction” activities that are required between the operating system and applications developers and safety analysts during the development lifecycle. These activities are needed to facilitate largely independent safety analysis of the components, whilst still ensuring that the system is acceptably safe. We then demonstrate how these techniques will help establish a maintainable safety case for IMA.

### Integrated Modular Avionics

The avionics standard ARINC 653 (ref. 3) contains specifications for an individual IMA computing module. The standard proposes a layered architecture as shown in Figure 1. Each software application is divided into a number of partitions. 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 protection of storage space for a partition is a particularly important function as applications of different development assurance levels (DALs) are likely to be running on the same module. The data/code of the high integrity (DAL) applications must be protected from low integrity (DAL) applications. The use of temporal partitioning guarantees each partition access to the processor for a fixed time slice per cycle. A number of modules will be linked via a common network e.g. ARINC 629 (ref. 4).

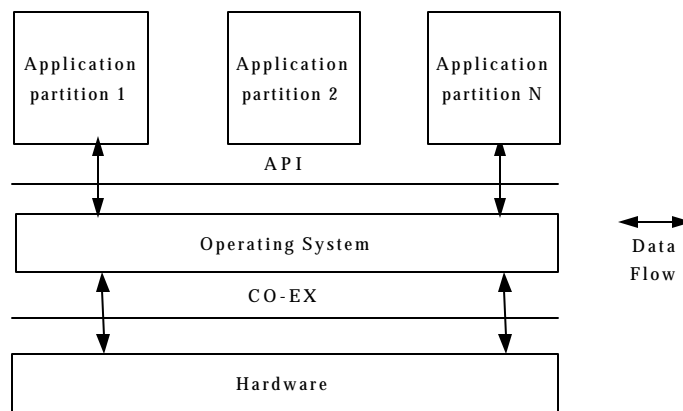


Figure 1 - IMA Computing Module Architecture

Applications request services, such as communications, via a defined Application Programming Interface (API) layer. The operating system responds to these requests and in turn requests hardware services via a Core-Executive (CO-EX) or hardware interface layer. The use of interface layers hides the underlying implementation from application components as far as is practicable. This means, for example, some aspects of the operating system may be changed without requiring any alteration of the applications. Also the hardware can be upgraded without alterations being required to the OS and applications (other than the requirement for recompilation of the code to the new hardware).

The use of logical partitioning also means that application partitions can be arranged in numerous different configurations on the modules so that they meet their requirements, e.g. for access to processing facilities and so that backup lanes are available. Theoretically, any application running on the IMA platform could be located on the same module as any other application with them sharing access to resources on that module. The ability to reconfigure the system means that in the presence of a computer hardware failure any affected applications can be run on a different module.

Unfortunately, at present, there is no specification for the CO-EX layer and the ARINC 653 specification is not complete (for example there are deficiencies in the health management specification (ref. 5)). However, even without a CO-EX the process of upgrading the computing hardware should require significantly less rework than at present as the application software will have no explicit interactions with the hardware. This is important, as hardware obsolescence is a significant problem for the aviation industry, with some federated systems hardware going out of production before an aircraft enters service, and with a typical aircraft lifecycle of 25-30 years. Some features of IMA have been introduced in commercial aircraft, such as the DEOS operating system (ref. 6). However, this was certified as a component of a complete system. The next section examines certification issues in more detail.

### Current Certification Practice

This section discusses current practice for safety analysis and certification in the civil aerospace industry and discusses the difficulties of applying this to IMA based computer systems. In general, the current standards assume that the computer systems are federated, that is each application runs on its own dedicated hardware unit and computer hardware failures are confined to largely independent physical units. This is no longer the case for an IMA system. Also, current analysis techniques examine software and hardware as an integrated whole, producing a monolithic safety case for each system. This type of approach will not allow the full benefits of IMA to be realised as it only works for a fixed configuration of partitions and modules. Any change would mean that all the analysis would need to be re-examined. This is potentially extremely time consuming and, in any case, prohibitively expensive. In addition the IMA infrastructure and applications are to be developed independently, meaning the analysis could only be performed at the end of the development lifecycle. If any problems were brought to light by the analysis both applications and IMA might have to be redesigned, and then re-analysed. Clearly some kind of compromise is required, which is based on current practice, but which supports independent development of the IMA infrastructure and the computer applications. An IMA variant has been proposed for the Airbus A380 and this provides the driver for this research.

The Aerospace Recommended Practice (ARP) documents 4761 (ref. 2) and 4754 (ref. 1) describe a number of procedures to be undertaken during the safety lifecycle. Firstly Functional Hazard Assessment (FHA) should be undertaken. This identifies failures at both aircraft level and system level. Failures are categorised according to their effect, measured both in terms of reduction of safety margins and on the effect on the crew workload. Each category has an associated safety requirement expressed in terms of a permitted occurrence rate measured per flight hour. A system is assigned a DAL based on its most severe failure condition classification. DALs range from A to E. A system which has a potentially catastrophic failure condition is assigned DAL A. The DAL also indicates the degree of rigour in testing and analysis of the system.

A Preliminary System Safety Assessment (PSSA) is then undertaken. This examines a particular proposed system architecture and should identify and capture derived safety requirements. Analysis undertaken in the PSSA includes the use of techniques such as fault tree analysis to determine how potential function failures identified in the FHA may occur (i.e. to identify causes). Finally, a System Safety Assessment (SSA) is undertaken to verify and validate that system safety requirements have been met.

Common Cause Analysis (CCA) is also required. This consists of three analysis techniques used either to demonstrate independence or to identify where dependencies between functions exist. Zonal Analysis examines the layout of systems to determine if a fault at a particular location can affect independence, e.g. a hydraulic failure can affect a supposedly independent electrical system in the same vicinity, for instance, by leaking hydraulic fuel into it. Particular Risks Analysis examines whether systems are vulnerable to an isolated environmental event, e.g., a bird or lightning strike. Common Mode Analysis verifies that ANDed events in a fault tree are truly independent or if they might fail in the same way (mode) under given conditions. It includes examination of failures such as design flaws in hardware or software.

DO-178B (ref. 7) contains guidance on the use of software in civil avionics systems. Although it is not actually a standard it is generally used as part of the certification and development lifecycle for avionics software. Software is assigned a DAL according to the most severe failure condition it can contribute to or cause. The techniques used in development depend on the assigned DAL. Level A software has the most rigorous requirements placed upon it. It might be thought that the IMA platform could just be developed using the level A processes, and that this would resolve the problems identified above. However, this would only show that a set of guidelines have been met, not that the IMA software is fit for purpose or that it will be acceptably safe in context. The “fault” is not with DO-178B, but the lack of safety requirements for the IMA platform. DO-255 (ref. 8) contains guidance for design of Avionics Computing Resources (ACRs), i.e. IMA modules, and is based heavily on the ARINC 653 standard. This document contains tables of information which should be provided by the IMA platform supplier, such as communications times and hardware response times. Whilst this information is clearly useful as part of a safety case, the document does not describe how to analyse the system in context, and does not provide safety requirements for IMA.

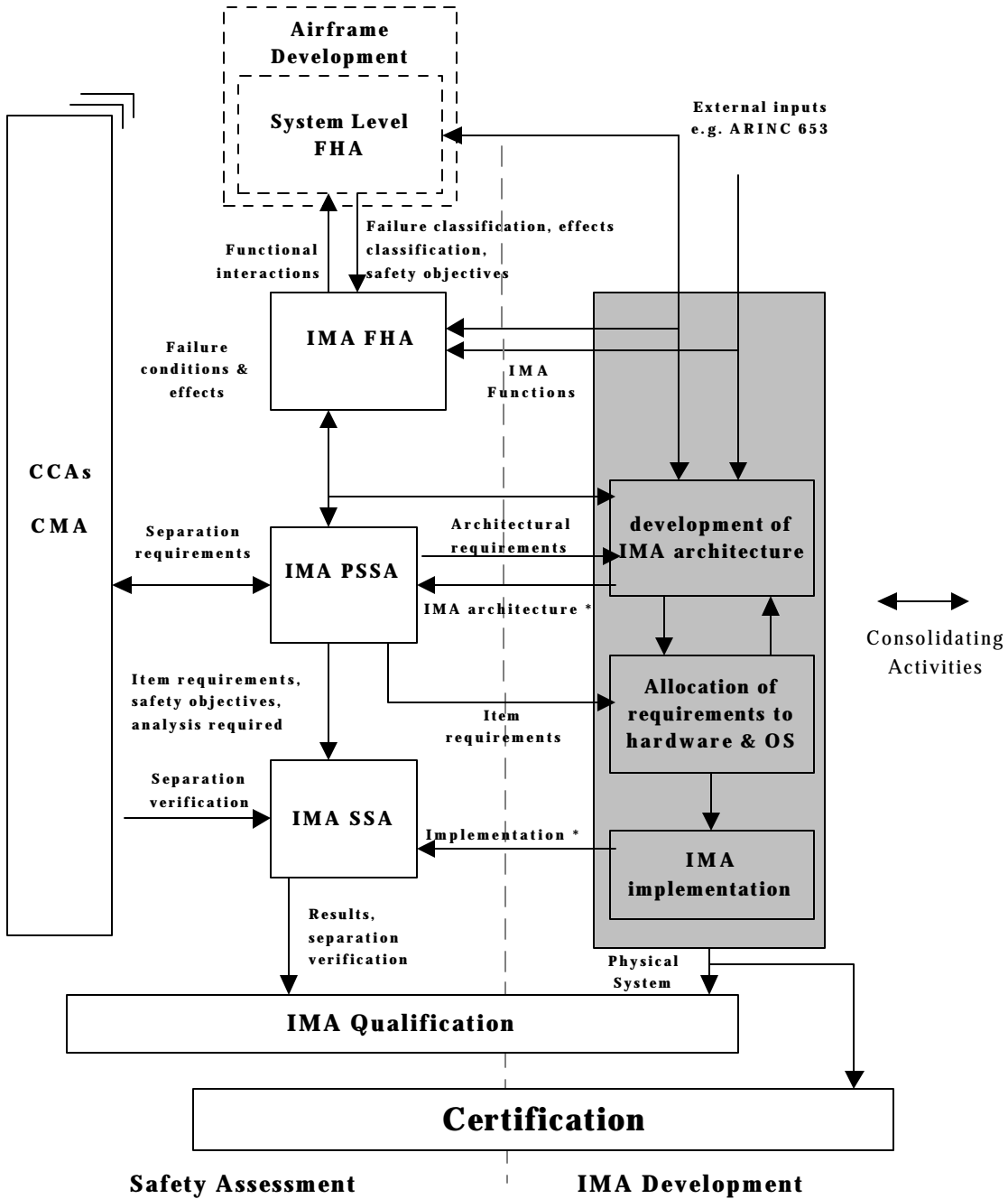


Figure 2 – IMA Certification Strand

Certification of IMA Systems

The following section describes an approach to certifying IMA, using three separate process strands. The three strands reflect the aircraft level (processes to be undertaken by the airframer), the IMA Development level (processes to be undertaken by the IMA supplier) and the application development level (processes to be undertaken by each application supplier). The process model has been adapted from ARPs 4754 and 4761. The strand for IMA only is shown for reasons of space (Figure 2). The usual activities, discussed earlier, from the ARPs are undertaken for all three strands but with a few alterations. The system level safety assessments performed by the airframer will be less detailed, with the bulk of the detailed analysis being undertaken by the IMA and application developers. The airframer will instead concentrate on the integration

and reconciliation of the outputs from the other strands of activities. The CCAs will need to be extended to examine the impact of total and partial loss of IMA in order to assess whether backup systems outside of the IMA system are required. This analysis will also assess the impact of the loss of various combinations of applications as this may have implications for the applications configurations. The IMA CMA will need to be extensive as the use of a common platform is likely to increase the risk of a common mode failure. More details of the aircraft and application process strands can be found in reference 9.

In addition to the individual process strands a number of levels of “consolidating interaction” activities are undertaken between the IMA suppliers and the application suppliers. These are parallel to the development process and help ensure that the overall safety objectives will be met when the individual components are integrated. The first level of activity examines requirements at a high level, looking at the functionality an application will require from the operating system in order to meet the application’s aircraft level functions. At this level a preliminary FHA will be undertaken for both the IMA infrastructure (i.e. the OS and hardware) and applications. The second level is the architectural level. This looks at the high level design of the IMA infrastructure and the constraints this will make on the applications. For example, requirements for application partition layouts and network links to sensors and actuators will be produced. The output of the IMA PSSA also provides input to the application PSSAs, (e.g. for bottom level fault tree events).

Behavioural issues are considered at the next level. This includes details such as the OS’s response to a failure and the flexibility and type of available recovery actions. Finally, the fine details are examined at the performance level, looking at, e.g., the timing characteristics of the communications system. The consolidating activities at each stage of development are shown in Figure 3. The information generated and required for each activity is detailed in Table 1.

Table 1 – List of Consolidating Interaction Activities

Level	Name of Consolidating Activity	Data	Direction of Flow
1	API requirements, IMA constraints, IMA functions and services	<ul style="list-style-type: none"> <li>• API requirements</li> <li>• Output of IMA FHA</li> <li>• List of IMA functions and services</li> </ul>	IMA to Application
	Application functional calls to IMA	<ul style="list-style-type: none"> <li>• Definition of functions of applications in terms of IMA functions needed to meet aircraft functions</li> </ul>	Application to IMA
2	IMA architectural constraints	<ul style="list-style-type: none"> <li>• Constraints or conditions related to IMA architecture</li> <li>• Output of IMA PSSA</li> </ul>	IMA to Application
	Application architectural calls to IMA	<ul style="list-style-type: none"> <li>• Evidence that application architecture fulfils the conditions imposed (e.g. architectural constraints, services to be called) in order to meet the safety requirements of IMA</li> </ul>	Application to IMA
3	IMA behavioural constraints	<ul style="list-style-type: none"> <li>• Relates to predicted behaviour of IMA during operation, e.g. <ul style="list-style-type: none"> <li>○ configuration</li> <li>○ partition mapping</li> <li>○ fault handling or</li> <li>○ provision for health monitoring</li> </ul> </li> </ul>	IMA to Application
	IMA behavioural constraints fulfillment	<ul style="list-style-type: none"> <li>• Describes how applications respond to/meet IMA behaviour</li> </ul>	Application to IMA

4	IMA performance constraints (after implementation)	<ul style="list-style-type: none"> <li>Resource allocation, WCRT, etc.</li> </ul>	IMA to Application
	Application process calls to IMA	<ul style="list-style-type: none"> <li>Evidence that applications either: <ul style="list-style-type: none"> <li>meet existing constraints, or</li> <li>require specific extra resources</li> </ul> </li> </ul>	Application to IMA

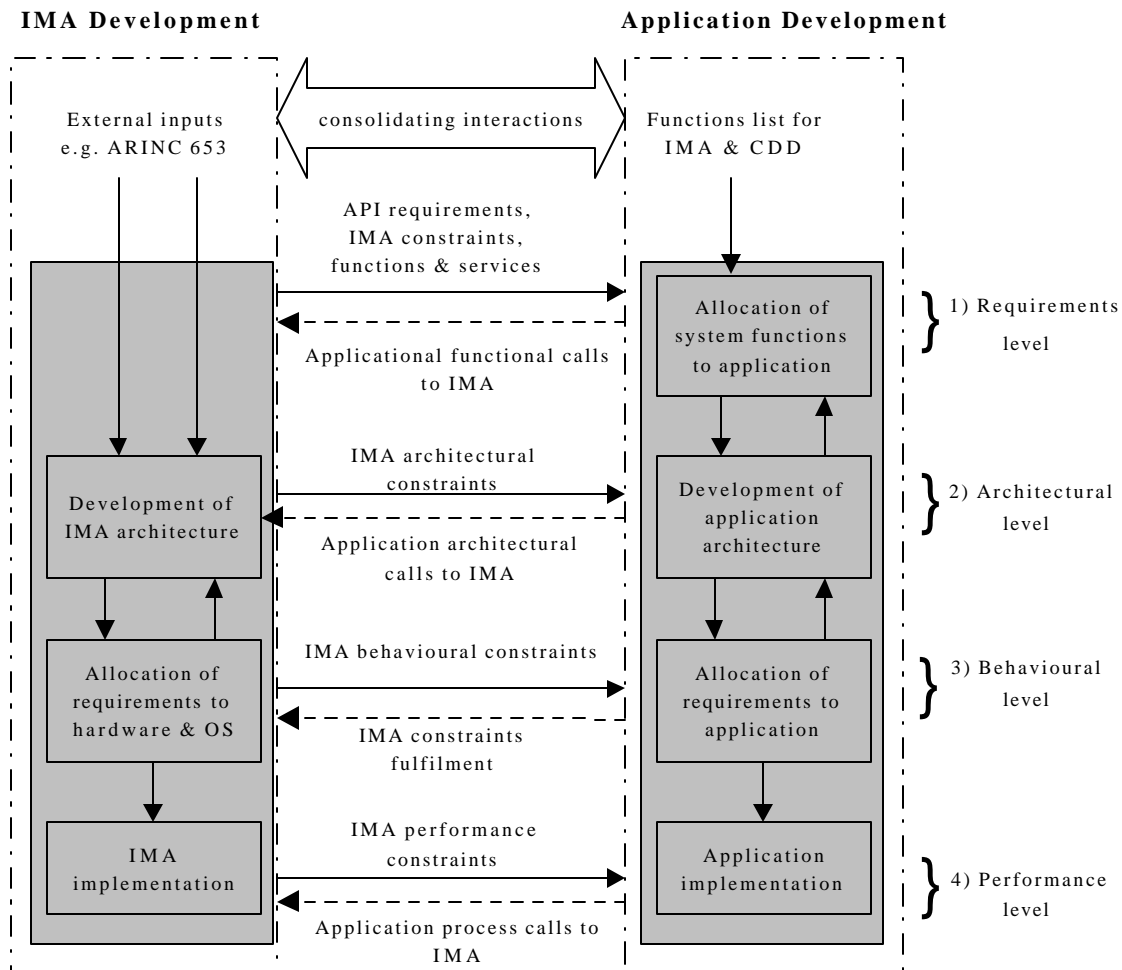


Figure 3 – Consolidating Interaction Activities between IMA Supplier and Application Suppliers

### Building a Safety Case using Contracts

At each level of the consolidating activities data about the interactions of components is generated for use as part of the overall safety case. This data can be captured using a technique such as “contracts”. The contract philosophy comes from Object Oriented Design (ref. 10) where the term is used to describe the relationship between two components, where one is a client and one is a supplier. The client component requires a service from the supplier but in order to obtain this service it must satisfy a number of pre-conditions (e.g. types of input). The supplier will then satisfy a number of post-conditions to provide the service (for example, a number of values may be recalculated and updated). This section describes an extension to this approach for use as part of a maintainable safety case, using the data from each level of consolidating interaction.

A safety contract defines the relationship between two or more components within a safety critical system. Contracts will be multi-part, reflecting the levels of consolidating interactions in Figure 3. For each contract there will be at least one supplier and at least one client. The simplest case is a contract between a single client and a single supplier (for example, a piece of software and a dedicated actuator). There will also be contracts between a single client with multiple suppliers (such as a fuel system may have two mechanistically different methods for determining the amount of fuel remaining) and contracts between a single supplier with multiple clients (such as an operating system servicing multiple applications). Note that, in the latter case, from the clients' perspective there is a one-to-one relationship, but from the suppliers point of view there is a one-to-many relationship. Satisfaction of contracts will form key elements in the safety case. As an example of this approach we now consider memory partitioning for an IMA system.

As described previously, each application on the IMA system will be divided into a number of memory partitions. A partitioning mechanism is then used to detect and prevent any attempts to access memory outside of an application's boundary. It should be noted that the effects of an actual violation are extremely difficult to predict (particularly in a reconfigurable system) as they will depend on which data has been altered. It could have no effect at all, or alter a critical parameter causing a potentially catastrophic failure condition. The partitioning mechanism should also have a short response time as a number of the tasks on an IMA system will have hard real time deadlines. A typical claim required as part of the safety case for any application which has potentially hazardous failure conditions arising from data corruption would be that the application has "timely and secure access to memory".

Two alternative methods for memory partitioning are Software Fault Isolation (SFI), which actually alters an applications low-level code to ensure that it cannot access memory outside it's designated area (ref. 11), and a hardware based solution using the Memory Management Unit (MMU) found on all modern processors (ref. 12). The MMU can be used to police the memory accesses of applications and assess whether the attempted access is valid. This approach has the advantage that all applications are automatically protected by the IMA infrastructure. To analyse a memory partitioning system using current guidance and techniques would require a fixed memory map of the applications locations and an examination of all their interactions. As stated previously this approach is not practical. The applications do not in fact contribute to memory integrity at all; rather they rely on the IMA platform to ensure their memory area is protected. This is a client/supplier relationship (contract), where the application is relying on the IMA platform to ensure data integrity. However, in order to benefit from this service the application will need to meet certain constraints imposed by the IMA platform. The safety contract can be used to detail the requirements and constraints generated at each level of consolidating interactions.

At the first level of detail (equivalent to the first level of consolidating interaction) a memory partitioning contract will merely be a statement of requirements. That is, the application relies on the IMA platform to "provide timely and secure access to memory" (see above). At the architectural level the contract will state the type of method to be used e.g. SFI or hardware based. For this example an MMU based system has been used as a solution. This places a constraint on the application, namely that it will be comprised of one or more partitions. This will be important for the PSSA of an application which looks at whether the architecture can meet overall system safety requirements.

At the behavioural level aspects of the system such as the OS's response to an attempted memory violation will be examined, for example will the application be shut down and any additional backup lanes be informed. Finally, at the performance level the actual values associated with the system will be given. Evidence about the client and supplier will demonstrate that each constraint can be met and will further determine whether the contract is met, and hence whether the overall claim is met in the safety case. The contract merely summarises the relationships.

A simple example contract based on the MMU memory partitioning solution in reference 12 is shown in Table 2. The example client is assumed to be an application with hard real-time requirements. Note that the client and supplier are logical elements of the system, rather than physical. The memory partitioning system is based on the PowerPC 603e and uses special features of the processor to supply both hard real-time

applications, and more dynamic applications, with secure memory storage. Specifically, special registers are used on the PowerPC processor which can hold a complete range of memory addresses, ensuring no data needs to be retrieved during a process execution, but limiting the application size to less than 4096K. It should be noted that not all processors have these features and the hence the performance/syntactic (and possibly behavioural) levels of the contract may be affected by another choice of processor. In practice, we would expect “high level” contracts such as that in Table 2 to be supported by more detailed specifications e.g. actual worst case response times at the performance level.

Table 2 – Example Proposed Memory Partitioning Contract

<b>Contract:</b>		
Client: Application X – Supplier: IMA Platform		
<b>Specification/Claim</b>		
Application requires secure and timely memory access		
<b>Architectural</b>		
Client requires	Supplier service	Constraints
Secure and timely memory access	Will provide hardware supported partitioning mechanism	Application will be divided into software partitions
<b>Behavioural</b>		
Client requires	Supplier service	Constraints
Secure memory space	Will detect a memory incursion	Applications which attempt to violate boundaries will be shut down. Service available from OS to inform specified backup partitions.
Access to memory space within bounded response time	Will provide service to execute with fully mapped memory	Application size will be limited.
<b>Performance/Syntactic</b>		
Client requires	Supplier service	Constraints
Secure memory space	Will detect a memory incursion with acceptable random failure rate	Applications which attempt to violate boundaries will be shut down. Service available from OS to inform specified backup partitions.
Access to memory space within bounded response time	Provides service with fully mapped memory using PowerPC BAT registers	Application size will be limited to < 4096K

It is anticipated that a sizable set of contracts will emerge during the development of the baseline safety case and some method will be required to ensure their completeness and consistency. This forms part of our ongoing research activities. However, once the safety case baseline has been established then the advantage of using contracts can be further demonstrated by considering upgrades to the IMA system. There are two types of change, an alteration to a client or to the supplier. In the case of the memory partitioning example this translates to an application or IMA infrastructure alteration. If a client is altered or a new client introduced then analysis is only required to show that the constraints are acceptable and satisfied for the client. This means that when an application is updated or inserted on to the IMA platform only the constraints within the relevant contracts have to be examined. The IMA platform (supplier) doesn't need to be re-analysed. However, if these constraints are not acceptable then an alternative strategy needs to be found such as re-configuring the applications.

A modification to the supplier will mean the constraints need to be re-examined. One such alteration to the IMA infrastructure could be an upgrade to the processor. In this case some of the constraints may need to be altered. However, this may not affect all levels of the contract, depending on the extent of the change. For

example, if a new type of processor is inserted, but an MMU is still used as a memory partitioning solution then only the behavioural and performance level constraints in Table 2 would need to be re-examined. This will limit the impact of the change within the safety case. Importantly, contracts make these impacts easy to identify and therefore vastly increase the speed with which the effects of a potential change can be assessed.

### Conclusions

An open distributed system such as IMA provides many potential benefits over existing aircraft computer systems, such as the ability to incrementally change components with limited impact on other components. However, to realise these benefits a new approach is required for safety analysis and certification. This paper has presented an approach based on the existing ARP 4754 and ARP 4761 civil aerospace process guidelines. This applies the ARP process model separately to the IMA platform and the applications but also uses a number of consolidating interaction activities at each stage of development to ensure that overall architectural and safety requirements can be met. These consolidating activities produce data to be used in a baseline safety case. Capturing this data in the form of a safety contract between a client and a supplier allows the impact of a change to be assessed and contained. A simple example contract between an application and the IMA platform was used to demonstrate how a contract might be specified for a memory partitioning system. We then demonstrated how the data captured in a contract would be used when altering either the client or supplier.

However, the contract approach still needs some refinement. In particular it is anticipated that a large number of contracts will be generated as part of the baseline safety case. A method is required to ensure the completeness and consistency of the set of contracts and to manage changes to them during the natural life-cycle of the system. There may be hidden interactions, where one supplier is actually a client to another supplier, meaning the extent of a change is greater than first anticipated. It would be desirable to limit these interactions based on knowledge of the types of change which may be applied to a system. Also required is some assessment of the roles and responsibilities of the client and supplier. For example, if a new client cannot work within the supplier's constraints then how much flexibility should be expected from the supplier to ensure interoperability? The next phase of our research will examine these issues.

### References

1. Society of Automotive Engineers. ARP4754: Certification Considerations for Highly-Integrated or Complex Aircraft Systems, November 1994.
2. Society of Automotive Engineers. ARP4761: Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment, December 1996.
3. ARINC, ARINC 653: Avionics Application Software Standard Interface. January 1997.
4. ARINC, ARINC 659: Backplane Data Bus. December 1993.
5. Conmy, P. and McDermid, J. High level failure analysis for Integrated Modular Avionics. in 6th Australian Workshop on Industrial Experience with Safety Critical Systems and Software. June 2001.
6. Dong, L., *et al.* Implementation of a Transient Fault-tolerance Scheme on DEOS. in Proc. of The Real-time Technology and Application Symposium 1999.
7. RTCA-EUROCAE, DO-178B/ED-12B: Software Considerations In Airborne Systems and Equipment Certification. December 1992.
8. RTCA-EUROCAE, DO-255: Requirements Specification for Avionics Computer Resource (ACR). June 2000.
9. Purwantoro, Y., Nicholson, M., and McDermid, J. A., Certification consideration of Airborne Open Computer Systems. Submitted to Special Issue of IEEE Transactions on Computers on Reliable Distributed Systems, 2003.
10. Meyer, B. Applying Design By Contract. in Computer. 1992.
11. Wahbe, R., *et al.* Efficient Software-Based Fault Isolation. in Operating Systems Review (USA) Fourteenth ACM Symposium on Operating Systems Principles. 1993.

12. Bennett, M. and Audsley, N. Predictable and Efficient Virtual Addressing for Safety Critical Real-Time Systems. in 13th Euromicro Conference on Real-Time Systems. 2001.

#### Biographies

P. M. Conmy, Research Associate, Department of Computer Science, University of York, Heslington, York, YO10 5DD, U.K, telephone - +44 1904-433385, facsimile - +44 1904-432708, e-mail – philippa.conmy@cs.york.ac.uk.

Philippa Conmy is a research associate in the BAE SYSTEMS funded Dependable Computing Systems Centre, at the University Of York. She attained an MSc in Computer Science from the University of Leeds in 1996. Prior to her current position she was a research associate at Leeds within the medical physics department. She is currently studying for a PhD at York, examining techniques for producing maintainable safety cases.

M. Nicholson, Research Associate, Department of Computer Science, University of York, Heslington, York, YO10 5DD, U.K, telephone - +44 1904-432789, e-mail – mark@cs.york.ac.uk

Mark Nicholson is a teaching and research fellow in the High Integrity Systems Engineering group at York. He attained his PhD in 1998 examining process allocation in real-time distributed systems. He has over 7 years of experience examining safety assessment of real-time systems. His current research interests include IMA certification processes and the evidence required for use of off-the-shelf operating systems in safety critical applications.

Y. Purwantoro, Research Associate, Department of Computer Science, University of York, Heslington, York, YO10 5DD, U.K, telephone - +44 1904-432792, e-mail – yudip@cs.york.ac.uk

Yudi Purwantoro is a research associate in the High Integrity Systems Engineering group at the University of York. His main research is concerned with developing certification procedures for IMA, particularly on the safety aspects, within the EU-funded VICTORIA framework. He is also a PhD candidate at the University of Sheffield. Prior to coming to the UK, he worked with the Indonesian Aircraft Industry as Certification Engineer

J. A. McDermid, Professor of Software Engineering, Department of Computer Science, University of York, Heslington, York, YO10 5DD, U.K, telephone - +44 1904-432726, e-mail – jam@cs.york.ac.uk

John McDermid has been Professor of Software Engineering at the University of York since 1987 where he runs the High Integrity Systems Engineering (HISE) research group. HISE studies a broad range of issues in systems, software and safety engineering, and works closely with the UK aerospace industry. Professor McDermid is the Director of the Rolls -Royce funded University Technology Centre (UTC) in Systems and Software Engineering and the BAE SYSTEMS-funded Dependable Computing System Centre (DCSC). He is author or editor of 6 books, and has published over 250 papers.