High level failure analysis for Integrated Modular Avionics

Philippa Conmy, John McDermid
Department of Computer Science,
University of York, York, YO10 5DD, U.K.
Email {philippa, jam}@cs.york.ac.uk

Abstract

Integrated Modular Avionics (IMA) is the term used for a common computer network aboard an aircraft. In order to gain full benefit from this technology a strategy is required to allow the separate development and safety analysis of applications and the computing platform. This paper presents the results of high level failure analysis of an IMA computing platform as a separate system and shows how the analysis can be used as part of an overall certification strategy for IMA. For the analysis six high level functions were constructed which described the functionality provided to applications and devices using the IMA platform. Lower level IMA services, such as scheduling and communications, are used to meet one or more of the functions. Deviations in service provision were considered using a number of guide words to suggest possible failure modes. The analysis revealed a number of weaknesses in the design which will require further consideration.

1 Introduction

Traditional aircraft computer systems are federated, with each system provided on a number of dedicated hardware units. Federated applications are physically separated from one another and analysis of the systems is undertaken individually. However, the aviation industry is moving towards the use of a common computing platform known as Integrated Modular Avionics (IMA). A full IMA system comprises of a number of computing modules communicating over a shared network. The basic platform supplies operating system services such as scheduling to applications running on the system. These applications may be spread across many modules and hence are not physically separated from one another. The IMA platform supplies mechanisms to ensure resources can be shared safely.

It is desirable to analyse each application and the basic IMA platform as separately as possible in order to allow their independent development and to facilitate reconfiguration and incremental upgrade of components. Unfortunately current certification standards and practice are still based on the federated approach and do not support independent analysis of applications on a shared resource. This is one of the reasons only limited use of IMA and IMA features occurs at present. New methods for certification and analysis are therefore required.

Copyright © 2001, Australian Computer Society, Inc. This paper appeared at the 6th *Australian Workshop on Safety Critical Systems and Software (SCS '01)* Brisbane. Conferences in Research and Practice in Information Technology, Vol. 3. P. Lindsay, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included

One approach to analysis is to independently examine the basic IMA services provided to an application such as scheduling and shared resource management. However, analysis of these features must consider the overall context of IMA in order to demonstrate safety adequately, and to show the affects of failures. This paper presents work being undertaken in collaboration with Airbus UK to develop meaningful analysis of the IMA platform as a separate system. The IMA specification used is taken from the international ARINC 653 civil avionics standard (ARINC 1997).

The layout of the paper is as follows. Section 2 describes current certification and analysis procedures and compares them with those needed for IMA. An introduction to the ARINC 653 model is given. Section 3 looks at the overall context of IMA, describes a set of high level IMA functions and links these with supporting lower level IMA services. Section 4 describes the analysis process undertaken for each IMA function. Section 5 presents some of the results of the initial analysis, and discusses several of the arising issues, including some revealed weaknesses in the ARINC 653 health management specification. Section 6 shows how this analysis fits into the overall IMA certification strategy. Finally a summary is given in Section 7.

2 Integrated Modular Avionics

Certification standards and analysis techniques used for computer avionics systems are largely based on the traditional approach of using federated computer systems. Federated applications run on a number of dedicated hardware units. Within these units software is generally tightly coupled to the underlying hardware. As each system is physically separated much of the safety analysis is undertaken independently. Analysis includes assessment of the effects, likelihood and severity of system failures as well as low-level analysis of software/hardware interactions, for example to demonstrate a task's Worst Case Execution Time (WCET) on the target hardware. Higher level analysis is undertaken to examine the effect of a system failure mode, or the effect of a combination of failure modes at the aircraft level. Again, this analysis relies on the fact that systems and their failures are largely independent. The depth of analysis undertaken depends on the Development Assurance Level (DAL) associated with a particular system. The DAL is allocated depending on the potential criticality and risk associated with a system failure. For example, the civil avionics software guidance document DO-178B (RTCA-EUROCAE 1992) recommends software be given a DAL ranging from A-E depending on the severity of its incorrect functioning.

Level A is allocated to software whose incorrect functioning can contribute to a catastrophic event, whereas Level E is allocated to software whose incorrect functioning has no operational effect.

There are a number of problems with the federated approach. Firstly, the lifetime of an aircraft (typically 25-30 years) means that the underlying computing hardware becomes rapidly obsolete, often before the aircraft is in service. Replacing out of date hardware is expensive, but upgrading the software to run on new hardware is also difficult due to the tight hardware/software coupling. Any alteration to the software will require re-analysis of the entire system and its interactions. In addition to this the many different systems use many different types of hardware; therefore a large number of spares are required.

To combat these problems it has been proposed that aircraft use a common IMA platform for all computer applications. IMA is the term used for a distributed real time network of computer modules, upon which the many different applications can run. Applications running on IMA must share access to resources such as processing and memory, and mechanisms are required to ensure this can be achieved safely. These applications only have access to common services such as communications via a defined Application Programming Interface (API) layer. Using the API should ensure applications are portable to new hardware.

One architecture for IMA is described in the civil avionics standard ARINC 653 (ARINC 1997). This architecture uses the concept of partitions to manage shared resources. 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 DALs may run on the same module. Therefore the data/code of the high integrity applications must be protected from lower integrity applications which may accidentally or deliberately try to access another partition's memory area. The use of temporal partitioning guarantees each partition access to the processor for a fixed time slice per cycle.

The IMA platform consists of a number of modules grouped in cabinets throughout the aircraft. These modules are networked to each other and to various Input/Output (IO) devices on the aircraft. The modules have a layered architecture as shown in Figure 1. As well as an API layer there is a Core-EXecutive (CO-EX) or hardware interface layer between the module operating system and supporting hardware to provide further hardware transparency. All layers beneath the API are hidden from an application. At present no ARINC standard exists describing the CO-EX interface but the API layer is described in detail in ARINC 653.

The IMA architecture provides a number of potential benefits. Firstly, as applications are hardware transparent the underlying hardware can be upgraded or different types of hardware can be used[1] without affecting the application software (providing the API layer remains the same). Secondly, an application can also be upgraded or incrementally changed without affecting other applications providing the partitioning is robust. Thirdly, the applications can be re-configured to any module within the IMA platform. This is useful if, say, a hardware module has completely failed. Clearly, in each case it must be demonstrated that application resource requirements can be met.

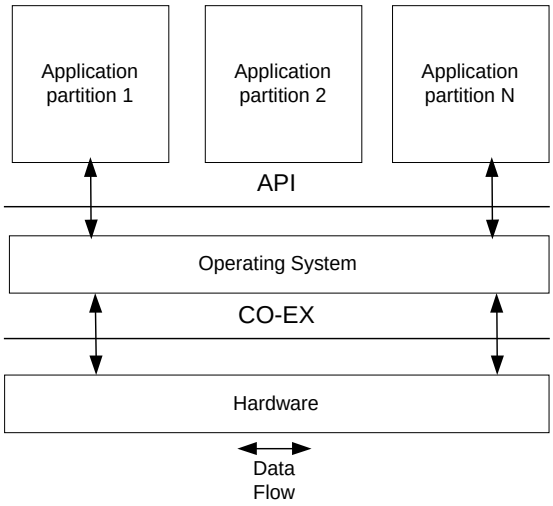

Figure 1: ARINC 653 IMA Module

Some features of IMA have already been used in commercial systems. For example the Honeywell Digital Engine Operating System (DEOS) (Ghosh, Heimerdinger et al. 1999) which has been used as a component aboard certified commercial aircraft, implements the ARINC 653 concept of partitioning and has a basic API. However, current certification standards and practices are still largely based on the federated approach and require examination of all internal interactions and software to hardware mappings. Issues such as logical partitioning are not addressed as means to support independent assessment. For example DO-178B (RTCA-EUROCAE 1992) only discusses logical partitioning as a means to separate defined sections of a specific application. This means that to certify an IMA style system, analysis must be undertaken for a specific configuration within which individual elements such as the operating system, API layer and applications must be examined as a single, complete unit. This has the disadvantage that analysis can only be attempted after the development of all applications and the IMA platform. Any issues arising from the analysis could mean all elements of the system require re-work. Also any alteration in system configuration may require extensive rework to support the system safety case. These certification difficulties are one of the reasons that only limited use of IMA and IMA features has occurred so far.

[1] For example to avoid a common design flaw in hardware which could cause the simultaneous loss of all modules.

To fully utilise the potential benefits of IMA, a certification strategy is required which allows the separate analysis of applications and IMA platform. This kind of strategy will also facilitate parallel development of these elements.

This paper presents high level failure analysis of IMA, without reference to specific applications, but using high level functions derived by considering the generic functionality which applications and devices using IMA require.

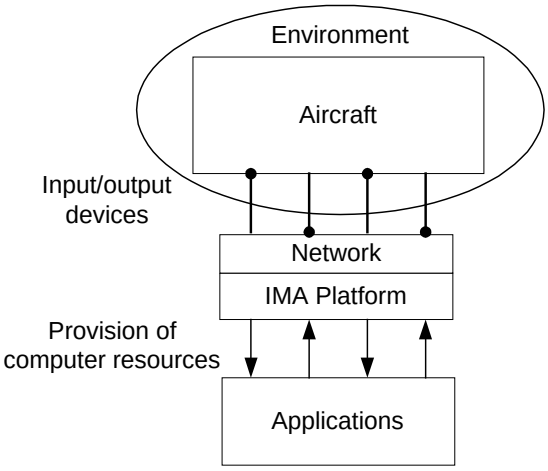

Figure 2: IMA Interactions

3 Context of IMA and high level functions

3.1 IMA Interactions

The IMA platform basically provides generic computing resources to the many different applications which run on the IMA modules. It also passes on information via the computer network to other modules and to various attached IO device, such as sensors and actuators. The relationships between these elements are shown in Figure 2. There are three basic failure categories associated with these elements:

1. incorrect functioning of the correct application/IO devices,
2. incorrect response to an application or IO event by the IMA system e.g. incorrect interpretation of instructions by the IMA system or failure to provide protection from insecure/failed applications and
3. inadvertent function of the IMA system i.e. a failure initiated by an error in the IMA platform with no prompt from an application or external device.

Evaluating certification targets, such as failure rates per flight hour, is complex for IMA, as the consequence of a failure within the IMA platform will depend on which application it happens to affect, and what mechanisms a particular application has to cope with failures. This is discussed more fully in sections 5.2 and 6. The analysis undertaken so far concentrates on what failures can occur within the system, and looks at what mechanisms are in place to deal with such failure rather than the potential risk or hazards which it could lead to.

3.2 IMA functions

For the analysis a set of six high level functions has been constructed based on the basic computing resource requirements of communications, processing and memory, and on the execution and architecture life cycle of IMA (see Table 1).

Initial attempts at analysis looked at the IMA service level (see section 3.3), but this was found to be unrevealing as there was often no obvious way to link the effect of a failure to an application effect, or to find an appropriate mechanism to deal with the failure. The IMA functions are described in detail below.

The first function concerns communications. Communications channels must be secure so that only applications or IO devices with specific permission can use them. The channels will have performance requirements for timely arrival and assurance of data integrity. These requirements are summarised in function 1, *Provision of secure and timely data flow to and from applications and Input/output devices.*

An application also requires access to a processing resource. The access must be managed so that timing deadlines can be met for all applications. This will be achieved partly by an applications use of basic IMA scheduling services but also by the use of configuration management and time partitioning to guarantee an application full access to the processor within a time slot. This is summarised in function 2, *Controlled access to processing facilities*.

The third required resource is memory. A partition's memory storage must be protected from corruption both from hardware failure and from external partitions. The latter is particularly important if memory space is to be shared between partitions with code of different DALs. The memory requirement is expressed in function 3, *Provision of secure data storage and memory management*.

The fourth function, *Provision of consistent execution state*, concerns consistency of data throughout the IMA modules. Application data will be loaded into modules from a data store, either for normal initialisation or following an upgrade or alteration. This data must be consistent across modules both in terms of version e.g. after an incremental change, and also in terms of correct numbers of partitions e.g. including backup lanes. There may also be additional flight data loaded at initialisation which should also be consistent and correct.

The fifth function, *Provision of health monitoring and failure management*, covers partial and controlled failures of the IMA platform. The IMA platform provides a number of health management (HM) services. Some of these are used only by the IMA services, for example to detect a hardware failure (there should be no hardware dependencies in the application software). Other HM services are used by an application, for example to report a failure to the IMA Operating System (OS).

No	IMA function	Specific example of potential failure
1	Provision of secure and timely data flow to and from applications and Input/output devices, e.g. sensors and actuators	Transmission of deploy command to landing gear in flight
2	Controlled access to processing facilities	Omission in schedule of fuel system pumping control partition
3	Provision of secure data storage and memory management	Corruption of fuel system execution code by lower integrity partition
4	Provision of consistent execution state	Incorrect or inconsistent flight data is loaded into system
5	Provision of health monitoring and failure management	Fuel system partition shut down when no error has occurred
6	General provision of computing capability	Total loss of IMA platform

Table 1: IMA functions

The defined actions to be taken by the OS in response to many failures are not yet fully defined and some of the HM failure types will be dependent on the hardware used e.g. for processing failures. Therefore the analysis of this function is incomplete at present (see section 5.2).

The final function is *General provision of computing capability*. This function is included to consider uncontrolled failures within the IMA platform such as a complete power loss and thus loss of all modules.

3.3 IMA services

Each of the IMA functions is supported by one of more of the lower level IMA services. These services can also support more than one high level function. The service categories are based on ARINC 653 (ARINC 1997), and also on the ARINC 651-1 standard (ARINC 1999), which is a top-level design guide for IMA. The full list of services is shown in Table 2. The specific ARINC API calls can be conveniently grouped within these eleven services.

4 Analysis technique

Each of the functions from section 3.2 has been analysed to identify possible failure modes, consider their causes and identify derived requirements to deal with the failures. The analysis has been driven using the guide words (see Table 3) from the Software Hazard Analysis and Resolution in Design (SHARD) and Low-level Interaction Safety Analysis (LISA) techniques developed at the University of York (Pumfrey 2000), (McDermid and Pumfrey 2000). These techniques are used to provide safety analysis for safety critical computer systems. SHARD examines possible deviations in information flows between software elements, and LISA examines deviations in timed events and system resource usage. The guide words are used to suggest failure modes and cover service provision, service timing, and service value. For this analysis possible deviations or failures in high level system functions are examined.

The analysis is performed as follows. Firstly, a guide word is used to suggest a possible failure mode of a function. Then possible causes of that failure are identified, generally from within those IMA services which meet or affect the high level function, but also considering the three possible categories discussed in 3.1. Finally, consideration is given as to what mechanisms exist to deal with the failure. These mechanisms are not exclusively the responsibility of the IMA platform as only an application may detect and act on certain failure modes. Some arguments as to the actual risk of the failure occurring are also required, for example if a particular failure is not deemed credible. Although it is usual with this type of analysis to provide a judgement on the criticality or severity of the consequences of the failure this has not been attempted for the reasons discussed in section 3.1. Further discussion of this issue can also be found in section 5.2.

As an example of the analysis consider the high level function 1, *Provision of secure and timely data flow to and from applications and Input/output devices*. A possible omission failure (where a service is not provided at all) would be that a data packet does not arrive at a destination application expecting to receive data. Possible causes of this failure include an application failure at source meaning the data was not sent at all or a failure in the IMA communications service, for example if the data was routed to the incorrect destination. The first cause is an application specific failure i.e. there is no requirement on the IMA OS to provide a detection mechanism. However, IMA support facilities for the receiving application may be required in terms of failure reporting services.

Possible methods to deal with the second cause of failure are more complex and could involve interaction between applications and the IMA OS. The receiving application can use the failure reporting mechanisms provided to inform the platform that an error has occurred. This report can then be used by the HM services to diagnose the error and appropriate action can be taken. This action is dependent on the chosen failure management policy and the level of authority given to the IMA OS.

Number	IMA Service	Service Description	Related IMA Fns.
1	Data loading	This service allows data to be uploaded and downloaded on the aircraft, and to be loaded into a partition for normal execution and for restart after a failure.	4,5,6
2	Timing watchdog	The timing watchdog allows processes to have real time deadlines and can detect failure to meet these deadlines.	1,2,5
3	Intra-partition communications	This group of services allows data to be exchanged between processes residing in the same partition.	1
4	Inter-partition communications	These services allow data to be exchanged between two or more partitions and external IO devices.	1,5
5	Scheduling	Scheduling services allow priority based access for tasks within partitions to demand processor time. Partitions themselves are scheduled in a cyclic manner.	1,2,5
6	Processing	This is the provision of data/instruction processing from the central processing unit	All
7	Module BIT/HM	This service provides Health Management (HM), Built In Test (BIT) and data logging when a fault is detected. This service includes both error detection and recovery.	All
8	Initialisation	For complete system initialisation data is loaded into partitions using the layout in the current system configuration. This service is also used for error recovery.	All
9	Close-down	This service can close down both single application partitions and individual modules (e.g. following a failure).	All
10	Partitioning	This service describes the mechanisms used to allocate resources to partitions, and police resource usage. This is both for spatial partitioning (e.g. access to communications) and for data partitioning (to prevent illegal access to memory space)	1,3,4,5
11	Configuration management	The configuration of the IMA system describes the arrangement of applications and IMA support throughout the IMA modules and network that ensures each application will meet its certification requirements.	All

Table 2: IMA Services

Guide word	Meaning
Omission	A service is not provided
Commission	A service is provided when it is not required (a perfectly functioning system would have done nothing)
Early	A service is provided early in relation to an expected time frame
Late	A service is provided late in relation to an expected time frame
Value	There are two types of value error to be considered – Detectable/Coarse and Undetectable/Subtle

Table 3: LISA and SHARD guide words

Guide word	Deviation	Cause	Detection/Protection/Mitigation
Omission	Data is not sent to destination from source	IMA routes data to incorrect destination	Use source/destination tags on data and check their validity against the configuration table. If this deviation occurs it will also lead to a commission failure (see commission analysis). Receiving application can detect data is missing by comparing communication time stamps. Application HM system may report error.
		IMA fails to send data accepted from application to destination	Receiving application can detect data is missing by comparing communication time stamps. Application HM system may report error.
		Network infrastructure error	Network specific fault. Receiving application can detect data is missing by comparing communication time stamps. Application HM system may report error.
	Data is not sent to source	Source partition was not initialised within IMA system	Detected on system initialisation by checking configuration Look Up Table – see analysis of function 4.
		Correct application source partition is closed down due to module/cabinet close down	Backup system on different module/cabinet provides functionality of source partition. If whole application is missing (due to systematic error) backup system external to IMA may need to be used.
		Correct application source partition is closed down, due to internal partition error	Backup system (e.g. a different partition) provides functionality of source partition. If whole application is missing (due to systematic error) backup system external to IMA may need to be used.
		IMA module does not schedule source partition	Timing watchdog detects error (see analysis of function 2). Backup system (e.g. on different module) provides functionality of source partition. If whole application is missing (due to systematic error) backup system external to IMA may need to be used.
		Data is not received from source input device	Receiving application can detect data is missing by comparing communication time stamps. Application HM system may report error and request action.
		Internal source application error	Receiving application can detect data is missing by comparing communication time stamps. Application HM system may report error and request action.

Table 4: Example showing analysis of function 1: Provision of secure and timely data flow to and from applications and Input/output devices

Whatever the cause of the failure, the receiving application needs to have some means to detect that no new data (or no data at all) has arrived at its destination. This detection can be supported by the IMA communications service, e.g. by the use of time stamps on sampled data.

This example shows that more than one derived requirement can arise from the analysis of an individual failure. These requirements need to cover the full range of possible recovery actions to meet the full range of possible application requirements and policies.

5 Analysis results

5.1 Major analysis results

An example section of the analysis is shown in Table 4. The complete range of required mechanisms and arguments derived from the analysis are too numerous to present in full, so this section presents a summary of those which appeared most frequently.

The analysis found many of the potential failures can be dealt with using the current ARINC 653 design. For example many late and early faults can be detected by using provided API services for setting process deadlines and timed waits. The time partition scheduling mechanism will provide protection locally from processor monopoly by an incorrectly functioning application. Other modules which may be at risk, e.g. if an application is attempting to flood the shared network with erroneous data, can be protected by reporting the rogue partition to the health management system which can then close down the offending application.

Other mechanisms have been refined by the analysis. For example ARINC 653 requires that applications be configured so that their resource requirements and availability and integrity requirements are satisfied. From the analysis it became clear that an application with duplicate or multiple lanes should be arranged so that if a failure causing the loss of a module or cabinet occurred the backup systems would still be available. Note that this assumes there will be no dynamic reconfiguration in the first generation of full IMA systems.

The use of a global clock or module time synchronisation mechanism solves a number of difficulties. For example, ARINC 653 requires that time-outs be placed on sampled data provided via the communications system. However, within ARINC 653 the concept of time is local to a module, so data sampled from a different module may be consistently out of date by the time an application tries to access it if the modules are not synchronised. This may not be detected if the local time base has drifted.

5.2 Health management issues

Frequently the derived requirements were that an application or module HM would detect a failure and then take appropriate action. A number of issues arose when comparing the HM strategies from the analysis with those from the ARINC 653 model. Firstly, the level or type of HM responsible for detection of a failure was not always clear. Secondly, the responsibilities for choosing an appropriate action did not always reflect the extent of the effect of the failure. Thirdly the chosen action will depend on the context of the failure - for example the criticality of the affected application(s) - and guidance is needed on how to choose action strategies.

ARINC 653 proposes three levels of errors:

1. module level e.g. errors during partition switching;
2. partition level e.g. errors during process management;
3. process level e.g. illegal OS request.

These have been categorised as to where the cause of or initial occurrence of the failure appears. To a certain extent it is not important where the cause lies, only that it is detectable so an appropriate action can be taken. Each of these error levels has an associated HM table containing recovery actions dependent on circumstance. The system integrator[2] is responsible for the module and partition level HM tables, whereas the application suppliers define the process level tables.

The three levels of errors hide the range of potential impact of an error. For example, one error categorised as being at the process level is a memory violation, implying that the impact is local to the current partition only. However, a memory violation is only classed as an error if an attempt is made to access memory outside of a process' current partition. Another process level error is a hardware fault; clearly such a fault could affect any application on the module. As such it seems inappropriate that the application supplier should be responsible for defining a HM action when the impact goes beyond the partition boundary. Equally it seems inappropriate that the system integrator be responsible for producing a HM action where a failure may not propagate beyond the local partition boundary e.g. for a process management error.

Given the analysis results we propose that the error types should be defined by who is responsible for their detection and the associated action(s) should be defined by the extent of potential impact. There are four basic types of error:

1. functional application errors detected internally e.g. an incorrectly calculated value;
2. non-functional or behavioural application errors detected by the underlying IMA platform e.g. a timing overrun or memory violation;
3. computing hardware failures detected by the underlying IMA platform e.g. by parity checks;
4. external hardware failures which may be detected by a monitoring application with specific knowledge e.g. a sensor failure.

[2] The system integrator is defined in ARINC 653 as the person responsible for providing a valid configuration for the IMA platform and choosing error management policies

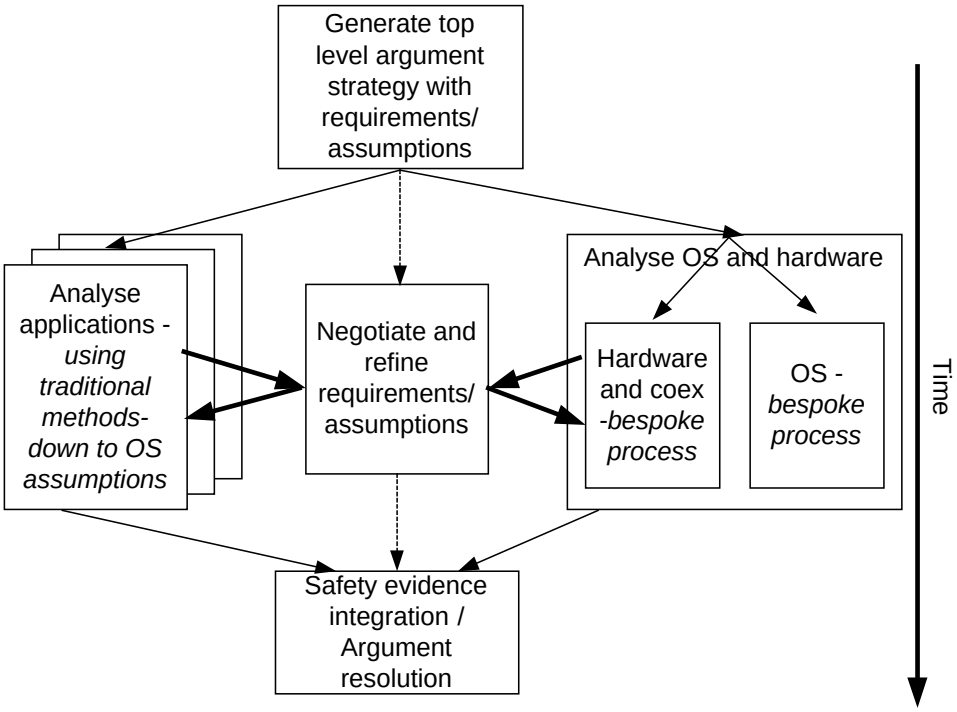

Figure 3: IMA certification strategy

For each detected failure an associated action (or set of actions) should be defined as part of a HM table. These tables can be customised for each application although standard sets could be used dependent on the application architecture e.g. dual-dual or command/monitor.

Devising the tables will require some interaction between application suppliers, the IMA platform provider and the system integrator. The actions taken may involve communication between the different levels of HM, for example to co-ordinate the close down of application channels on different modules

6 Overall IMA certification strategy

This analysis is only part of an overall IMA certification strategy shown in Figure 3. This figure clearly shows the separate development and analysis of applications and underlying IMA platform, with crossovers to negotiate and refine requirements and assumptions.

An example IMA safety argument can be found in (Nicholson, Conmy et al. 2000). This argument was structured to separate arguments about applications, their usage of IMA, and analysis of the underlying platform itself. The analysis presented in this paper represents work supporting the OS and API layer strands of the platform argument. Other IMA analysis will have to look at low-level issues such as processor verification. This cannot be undertaken until the hardware has been chosen and the CO-EX defined.

The high-level IMA analysis has raised a number of issues which will need to be examined as part of the negotiation process shown in Figure 3. This includes deriving HM strategies and also the criticality/ hazard risk assessment currently missing from our analysis. After this refinement the analysis can be taken forward to look at some of the issues such as HM in more depth. It is anticipated that some of the requirements for the CO-EX layer will be derived from the HM analysis, as this must consider generic hardware fault types.

Finally, the separate analyses will need to be integrated to form a complete safety argument for the IMA system and configured applications at the end of their development.

7 Conclusions

This paper has presented a high level failure analysis of IMA. The analysis was based on six basic functions required by other systems using the IMA platform. These functions are supported by lower level IMA services such as the API calls and partitioning methods. The analysis has revealed many derived requirements for the IMA platform including the need for further refinement of the health management system specification.

8 Acknowledgements

Philippa Conmy is a research associate in the BAE SYSTEMS funded Dependable Computing Systems Centre (DCSC) at the University of York. This paper presents work undertaken in collaboration with Airbus UK and Dr Mark Nicholson from the University of York. Mark Nicholson is supported by the Framework IV funded PAMELA (Prospective Analysis of Modular ELectronic integration in Airborne systems) project.

9 References

ARINC (1997). Avionics Application Software Standard Interface ARINC 653, Aeronautical Radio Inc.

ARINC (1999). Design Guidance for Integrated Modular Avionics, ARINC 651-1, Aeronautical Radio Inc.

GHOSH, S., HEIMERDINGER, W. L., LARSON, A., DONG, L., MELHEM, R. and MOSSE, D. (1999). Implementation of a transient-fault-tolerance scheme on DEOS. Proceedings of the Fifth Real-time Technology and Applications Symposium, Vancouver, Canada. pp. 56-67. IEEE.

MCDERMID, J. A. and PUMFREY, D. J. (2000). Assessing the Safety of Integrity Level Partitioning in Software. Proceedings of the Eighth Safety-critical Systems Symposium, Southampton UK. pp. 134-152. Springer.

NICHOLSON, M., CONMY, P., BATE, I. and MCDERMID, J. (2000). Generating and Maintaining a Safety Argument for Integrated Modular Systems. 5th Australian Workshop on Industrial Experience with Safety Critical Systems and Software, Melbourne, Australia. pp. 31-41. Australian Computer Society.

PUMFREY, D. J., (2000), The Principled Design of Computer System Safety Analyses. Ph.D. Thesis, Department of Computer Science, University of York.

RTCA-EUROCAE (1992). Software Considerations In Airborne Systems and Equipment Certification DO-178B/ED-12B., RTCA and EUROCAE.