

Approaches to Certification of Reconfigurable IMA Systems

Paul Hollow¹, John McDermid, Mark Nicholson
Department of Computer Science, University of York, York, YO10 5DD, UK
Email: paulh | jam | mark @cs.york.ac.uk
Tel: (+44) 1904 432789

Abstract. *The aerospace industry has been investigating integrated modular avionics (IMA) for some years. IMA offers greater flexibility in the use of computing resources by reconfiguring the software to employ different processors and communications, in order to recover from failure and to redistribute workload. Such reconfiguration offers benefits, but poses difficulties for certification since current certification practice requires assessment of each configuration.*

The approach we have adopted is to seek means of clearing a configuration of a system and to identify a number of “equivalent” configurations. This requires us to establish “safe” reconfigurations for the IMA system. Technically, we have formulated the search for a set of “equivalent” configurations as a multi-objective optimisation problem. Pragmatically, the search produces configuration tables which could be used by the IMA operating system to make a “safe” change to an “equivalent” configuration, when necessary.

INTRODUCTION

Conventional aircraft systems are *federated*, with each major function, or application, in a separate hardware unit. These units may be interconnected, but each is considered independently from the point of view of certification. This type of federated system can be seen in most current aircraft, with those of the A320's generation having perhaps 40-50 systems, and about 100 computers. These federated systems are expensive to develop and certify. They are also relatively inflexible, and give relatively poor resource utilisation. These limitations, amongst others, have led to research into an alternative approach known as *Integrated Modular Avionics (IMA)*.

The aim of IMA is to bring the flexibility of

distributed architectures, such as networks of PCs, to aircraft applications. With IMA a number of functions or application run on a processor, communicating via services provided by an operating system. At any time, a function will be allocated to a particular processor, but reconfiguration allows applications to employ different processors, and communications to use different media, e.g. buses, to respond to system load, failures or battle damage. IMA has many potential benefits including simplifying software upgrades, making it feasible to add new applications without changing the hardware, and assisting in meeting requirements for maintenance free operating periods (MFOPs).

The use of IMA changes systems in a number of ways. For example, there may be no physical boundaries between applications, and sensors and actuators may be interfaced to buses, rather than linked directly to the hardware unit where they are used. However, in this paper, our main focus is on the impact that the introduction of such IMA systems has on *safety certification*.

Under current certification procedures each system is assessed in isolation, and the certification process assumes that the software behaves in a deterministic manner. Applying this philosophy to IMA, it is likely that each possible architectural configuration would need to be analysed and certified separately. Applying IMA to an aircraft of the complexity of an A320 is likely to yield many millions of valid configurations. Thus, unless it is possible to find an alternative to the classical certification process, the cost of certifying different system configurations is likely to outweigh the benefits of IMA.

The objective of the work described here is to find ways of identifying a “family” of different IMA configurations which can be shown to be “equivalent”, and which can be certified by analysing only one representative member of the family. To achieve this we employ off-line analysis of different possible configurations, producing tables of “safe”

¹ Paul Hollow is a Research Student within the DCSC at York funded by EPSRC with a CASE award from BAE SYSTEMS.

reconfigurations. This supports reconfiguration under failure based on the tables produced during the off-line analysis. This basic approach should also provide pointers for certification of more advanced reconfiguration techniques.

In Section 2 we set the scene by considering the issues surrounding the certification of IMA systems and scope the problem addressed by this paper. In Section 3 we propose a process for identifying and analysing a group of equivalent architectures. In Section 4 we outline our heuristic search based approach to determining the group, provide details on the scoring of different proposed architectures and give some preliminary results. In Section 5 we return to the issue of the value of the approach to certifying IMA systems.

CERTIFICATION OF AVIONICS SYSTEMS

Accepted practices in the aerospace industry are to assess systems independently, for the purpose of certification. Recent international standards such as ARP-4754 (SAE 1996) and the accompanying ARP-4761 are intended to deal with “complex and integrated systems” for commercial aircraft. However they do not explicitly deal with issues such as IMA and, implicitly, they still reflect the “system at a time” approach to certification. Military standards such as DS 00-55 (MoD 1996a) and DS 00-56 (MoD 1996b) in the UK, and MilStd 882C (DoD 1993) in the USA are similarly mute on the subject of certification of IMA systems. Thus the standards do not help us when considering IMA, and we need to go back to more basic principles. However, it is first worth identifying in more detail the challenges posed by IMA.

For computer-based systems, certification currently relies on two main assumptions:

- system can be viewed, and assessed, as a single entity
- the behaviour of the system is *deterministic*

Neither of these assumptions is likely to hold for IMA systems due to their ability to reconfigure following failure. Even if reconfiguration occurs on the ground, the behaviour of the system isn't determined through its life. In fact the situation is even more complicated than this. Bradley (Bradley et al. 1998) lists seven areas of avionics system certification which are affected significantly by the use of IMA principles and technology:

- Isolation can no longer purely be provided by physically separating the system functions.
- Priority based scheduling (Burns 1995) will be required to support varying workload (this is non-deterministic, although it is predictable).
- 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 reconfigured, or 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.

Space doesn't allow us to investigate all these issues fully. In this paper we assume priority based scheduling will be employed. We concentrate on the issues of reconfiguration with the aim of allowing re-use of safety evidence. Note is taken of the other aspects so far as practical.

CERTIFICATION OF RECONFIGURABLE IMA

As we have seen the issues surrounding IMA are numerous and complex. We have focused on one particular aspect, reconfiguration, that we believe is amenable to systematic analysis via heuristic search techniques. We suggest that analysis of a reconfigurable IMA architecture, sufficient to support certification, can be achieved the following steps:

1. Model the system and allocate the software components to the given IMA hardware components to produce a “baseline” system. This “baseline” model must be shown to meet all performance and safety requirements¹.
2. Produce a list of systems that would result from any single hardware failure. That is, list every hardware architecture resulting from a single module, processor or link failure. Identify those systems that no longer meet their performance or safety requirements.
3. For those systems that no longer meet their performance or safety requirements after failure, reallocate the software components to produce a new hardware-software mapping which does meet the requirements (these are “equivalent” configurations). Produce a “reconfiguration table” of allocations for each new configuration².

¹ We assume that the architecture includes a level of redundancy. For example, if we required 30 triplex systems, we might employ 100 processors, giving the ability to lose 10 processors without degradation (assuming that communication is still available).

² The IMA operating system would use these tables at run-time to identify an appropriate change, to an equivalent configuration, in the event of failure.

4. If there remain failed systems for which there are no allocations that meet all performance or safety requirements then revisit step 1 to produce a new baseline model. Repeat steps 1 to 4 until an acceptable set of "equivalent" systems are found or a stopping criterion³ is met.
5. Investigate the *mode change* problem for each new configuration (see below).
6. Repeat steps 2 to 5 for larger numbers of hardware failures, up to the planned limit. Note that, beyond this limit, it may be possible to use configurations with degraded performance, e.g. by discarding non-essential functions.
7. Produce safety evidence for the baseline configuration, and evidence from the reconfiguration process that each "equivalent" configuration meets its requirements.

To be useful, this approach must be acceptable to the certification authorities. Whilst we have not discussed the approach in detail with the authorities it seems to us to be the smallest practical shift from current practice, whilst dealing with the complexities of IMA. We now discuss the means of realising this process, then return to the issue of acceptance by the authorities in the conclusions.

There is a choice between carrying out reconfigurations and mode changes in the air, or on the ground at the end of the flight. In the latter case, the exposure to further failures is relatively high, but there is no need to show that the mode change is safe. In the former case, which is more likely to be appropriate for military systems suffering battle damage, it is necessary to investigate the mode change time, and the risk of further failures or damage in this time. The models we develop below include mode change, but it would be simple to remove it to consider reconfiguration on the ground.

TECHNICAL APPROACH

For the purposes of showing the basic elements of our approach to reconfiguration we have produced a model of a simple IMA architecture. The aim is to extend and improve this model as work progresses to be more representative of industrial scale architectures. The model (Figure 1) consists of one IMA cabinet containing three line replaceable modules (LRMs). One LRM contains four processors, one contains three processors and the third two. All processors are fully connected using ADPM network links within the LRM. LRMs are fully interconnected within the cabinet via ATM point-to-point network links. Each LRM has a

"system" processor residing on the LRM and the network links external to each LRM.

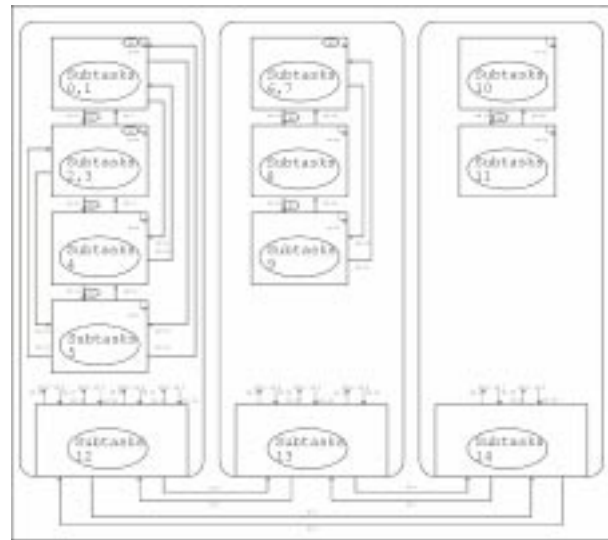


Figure 1: IMA Architecture Model

Three abstract tasks are defined to execute on the architecture. Each task is composed of a number of sub-tasks (processes). Each sub-task may generate one or more messages. The majority of the information contained within this architectural model remains constant under failure conditions, such as the memory capacity of processors and the requirements on the system. Some of the elements of the architecture however are amenable to change under failure conditions such as which processors are linked, the allocation of processes to processors and of messages to communication media. Thus we have an allocation problem; to allocate processes to processors and messages to communication media such that all performance and safety related requirements are met.

Extensions to this architecture can be easily envisaged such as multiple cabinets, heterogeneous processors within modules and restrictions on placement of software elements due to zonal issues, battle damage and failure. We are confident that the approach presented here can be adapted to incorporate such elements.

Determining an "Equivalent" IMA Architecture

Two IMA architectures can be said to be acceptable and equivalent if they both meet all the performance and safety requirements on the system. If a reconfiguration between two such architectures is to be acceptable then the *mode change* problem must also be addressed. Thus we define two IMA architectures to be equivalent under configuration if they meet all their requirements and a safe mode change path exists from

³ This may be a suitably low fitness value of the "baseline" system, or time spent on analysis.

one configuration to the other.

A safe mode change (done in the air) is one whereby the application code can be reloaded on the new processor, and data brought into step with other replicas of the application in a sufficiently short time that the likelihood of further failures giving rise to a hazard is acceptably low (Blackwell et al. 1999). In practice, the length of time needed for a safe mode change will depend on the characteristics of the application. Some applications may be virtually stateless, so the mode change time reduces to the code re-load time. In other cases there may be a large state, but it is fairly static, e.g. flight plan data. The worst systems will be those with large, dynamically varying, data where the new application will have to “play catch up” with a changing data set, e.g. information from sensors which deliver data at high rates.

In general, we can represent the notion of equivalence as a fitness value (function) which indicates how close to being equivalent two proposed configurations are. In fact, such a fitness function can be used directly to guide heuristic searches. At this stage in our work, we use a simple fitness function with four main elements:

- Resource usage - hardware and software constraints placed on the system.
- Timing properties - each task must be shown to meet its worst case response time (WCRT) deadlines.
- Time to Failure - each task must be shown to have an acceptable failure probability, expressed as the mean time to failure (MTTF) of the task.
- Costs of reconfiguration - cost of moving from one configuration to another must be within acceptable bounds.

If two IMA architectures are equivalent then their fitness values will be the same. In practice it is convenient to represent fitness by “penalties” for allocations (hardware-software mappings) which do not meet the requirements. A “basic” fitness function is:

$$P(s) = k_1 net + k_2 proc + k_3 task + k_4 subtask + k_5 mes$$

where

k_i = weighting factor for the element

net = penalty for too many messages on a communication medium, plus penalty for too many bytes of data, plus penalty for number of messages that should not be placed on the same communication media, for fault tolerance reasons, but are.

proc = penalty for processors with too many processes allocated, plus penalty for too much memory being used by resident processes, plus penalty for inappropriate

placement of system processes, plus penalty for number of processes that should not be placed on the same processor, for fault tolerance, but are.

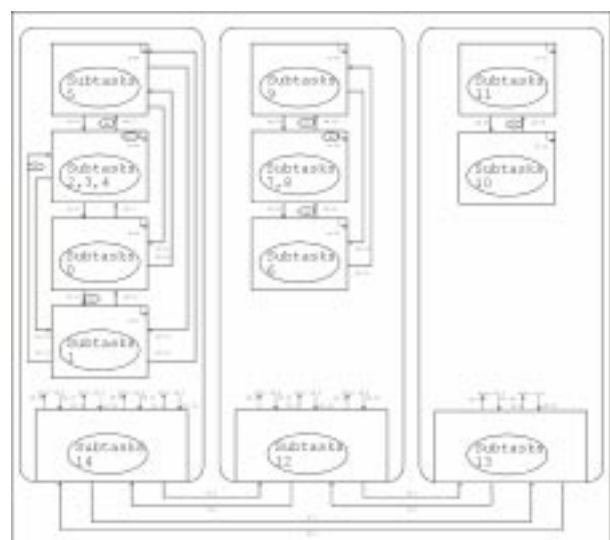
task = penalty for number of tasks that fail to meet timing deadlines plus penalty for tasks that fail to meet MTTF requirements

subtask = penalty for subtasks allocated to inappropriate processor (including failed processors), plus penalty for subtasks that do not meet their deadlines

mes = penalty for messages using a failed link, plus penalty for messages routed so that they cannot reach their destination process, plus penalties for any message that fails to meet its delivery deadline.

As can be seen, even this “basic” fitness function is quite complex, due to the need to represent elements for each major aspect of the system resources of the IMA architecture.

The primary reason for determining a set of equivalent architectures is to allow reconfiguration under failure. Suppose that the system is working in its baseline configuration and a processor fails. Suppose further that four processes reside on this processor. A new system configuration is required such that these processes are placed on different, working, processors. This new configuration must be “equivalent” in that it meets all the system requirements. Extensive changes may be required to facilitate this, such as changing priorities of processes, reallocating messages to different communication media and moving processes other than the four directly affected. In Figure 2 we show two reconfigurations under failure for the example presented in Figure 1; the first reconfiguration is optimal the second is not.



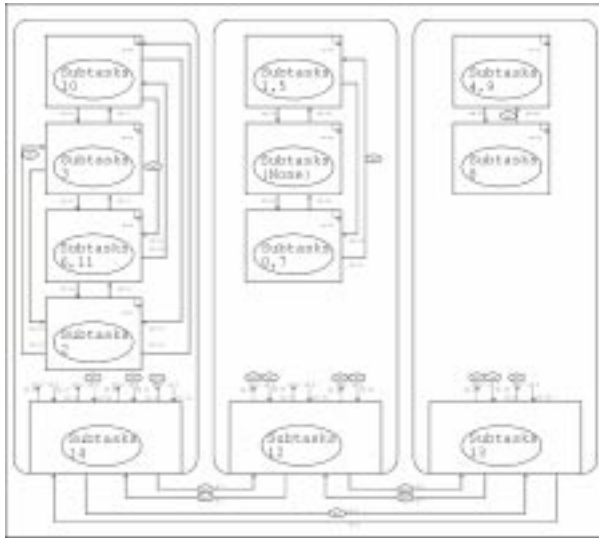


Figure 2: Two equivalent Architectures under Failure for Example Architecture

To include the elements of system reconfiguration and reliability and also the worst case response time, the fitness function is extended as follows:

$$P(s)' = P(s) + k_6 \text{reconfig} + k_7 \text{reliability} + k_8 \text{WCRT}$$

where

reconfig = penalty for time taken to reconfigure the system (minimise mode change time).

reliability = penalty for MTTF of tasks in the system (maximise overall MTTF).

WCRT = penalty for gap between WCRT and deadline (maximise to indicate timing flexibility in the system).

This extended fitness function is adequate for identifying “equivalent” configurations which meet basic performance and safety requirements, assuming that functionality is assessed independently, and is not impaired so long as the software has sufficient resources to execute.

In order to evaluate these parameters we require a number of systems and software engineering models. For example, we use standard reliability models, specifically reliability block diagrams (Sahner et al. 1995) derived from the hardware-software mapping, to evaluate reliability. As we are ultimately interested in safety, we will need to model the rate of occurrence or probability of hazardous failure conditions. At this stage, our concern is more with establishing a basic framework, so we have not employed fault trees or Markov models. (We have previously used Markov analysis in such applications (Nicholson 1998) so we are confident that we will be able to extend the approach, in due course.) Worst case response time

requires models of the worst case execution time (WCET) of the individual processes, and a global scheduling model (Burns 1995).

Searching for Good "Equivalent" IMA

In some circumstances it may be easy to define an acceptable equivalent architecture when a system failure has occurred. However, in general this will not be the case. For certification purposes we need to show that no single point of failure can lead to the system not meeting its requirements. Thus, we have to find $n+m+1$ equivalent architectures, where n is the number of processors and m is the number of communication media, if we are to show that no single hardware failure can lead to the loss of the critical services provided by the IMA system. Clearly, the greater number of failures we consider, the larger the space of possible configurations we have to analyse – however the problem is much worse than just finding a set of valid hardware configurations.

Only a proportion of the attributes of an IMA architecture are amenable to change under reconfiguration. These are the:

- processor a process resides on
- communication medium a message travels on
- priority of processes and messages

Setting the values of these attributes for a given IMA architecture is known as the Allocation Problem. We thus have to solve at least $m+n+1$ allocation problems. Each allocation problem can be formulated as an assignment type problem (ATP) (Chern 1992). The important features of an ATP problem, with respect to producing a solution for non-trivial sized problems, are:

1. checking whether a proposed solution is “acceptable” with respect to a given set of criteria can be undertaken in polynomial time.
2. finding the optimal solution, with respect to a given set of criteria, is non-polynomial (NP). That is, calculation and enumerative searches cannot in general be guaranteed to find an optimal solution in polynomial time.

Ribeiro (Ribeiro et al. 1994), indicates that an implicit enumerative (guided) search, or some other form of problem specific heuristic, is required in order to find good solutions for such problems within a “usable” time period. At this stage of our work, Simulated Annealing (SA) (Chern 1992; Kirkpatrick et al. 1983) has been chosen to address the problem of finding a group of equivalent allocations, although we anticipate that it may prove appropriate to use other heuristics at a later stage.

Simulated Annealing (SA) has been applied to a wide range of practical problems (Dowland 1994). As

its name implies, it models the behaviour of cooling substances, e.g. metals, that go through an annealing process. Consequently the algorithm is controlled by setting an initial temperature and a cooling rate. A simple sequential SA algorithm for a minimisation problem, is presented in Figure 3. Tools exist to implement this search algorithm. We use the X-Samson tool developed at the University of East Anglia (Mann 1997). The initial solution, s_0 , is chosen randomly, and therefore may or may not be a legal solution. The value $P(s)$ for each proposed IMA architecture is calculated using the fitness function presented in Section 4.1. The initial temperature, t_0 , is chosen automatically by the algorithm so that virtually all proposed moves to other possible solutions are taken. The temperature is reduced by a factor α each iteration, thereby reducing the number of moves that are accepted. As the temperature reduces, the set of solutions considered reduces, and the algorithm terminates with the result being the closest approach to a solution which meets the systems requirements, or an “optimised” solution if more than one is found which meets the requirements.

Problem: minimise $P(s)$ such that $s \in S$

```
Select an initial Solution  $s_0$ ;
Select an initial temperature  $t_0 > 0$ ;
Select a temperature reduction function  $\alpha$ 
Repeat
  Repeat
    Randomly select  $s \in N(s_0)$ ;
     $\delta = P(s) - P(s_0)$ ;
    if ( $\delta > s_0$ ) then
       $s_0 = s$ ;
    else
      generate random  $x$  in range (0,1);
      if ( $x < \exp(-\delta t)$ ) then  $s_0 = s$ ;
  Until iteration_count = nrep;
  Set  $t = \alpha(t)$ ;
Until stopping condition = true;
 $S_0$  is the approximation to optimal solution
```

Figure 3: SA Minimisation Algorithm

Each proposed solution represents a possible IMA architecture and is coded as an integer string of three sections; one representing the allocation of processes to

processors, one representing the allocation of messages to communication media and one representing the priorities allocated to processes and messages. Thus the length of the integer string is dependent on the number of defined processes and messages between these processes.

Once a baseline architecture has been produced by running this search algorithm for the scenario where all hardware items are working we can investigate reconfiguration under failure. We do this by making a single hardware component unavailable. We then rerun the search algorithm to find the best equivalent architecture that does not employ the failed component. To facilitate this we restrict the set of admissible allocations so that no software component can use this piece of hardware and we employ a minimum change metric in the form of a measure of the time it takes for a reconfiguration to complete. We then repeat this procedure for each single hardware point of failure. Clearly this approach can be extended to deal with multiple failures which can still be accommodated within the levels of redundancy provided by the IMA system.

If we find that there is no reconfiguration available for some failures, especially for single failures, then a more extensive redesign may be required. However, it is also possible that we failed to employ an appropriate baseline architecture. We may wish to revisit this architecture in the light of this new evidence. It is hoped that eventually we may be able to co-evolve the baseline and configurations under failure.

The results of this analysis are a baseline architecture and a set of reconfiguration tables that can be used by an IMA operating system to reconfigure a running system in the event of failure, knowing that the resultant configuration still meets performance and safety requirements. Figure 2 illustrates the results that can be achieved by this form of search⁴. It is our contention that the certification evidence for the baseline configuration and evidence that all the allowed reconfigurations are “equivalent” should be viewed as sufficient to clear the system.

Results to Date and Current Activities

We have analysed a number of simple example systems to develop the ideas and the analysis testbed.

By doing this, we have gained confidence in the soundness of the approach, as we have produced configurations which are amenable to manual checking. This gives us a suitable base for investigating industrial scale problems, which can’t be assessed manually.

⁴ Note that in this run the “minimum change” metric is not employed.

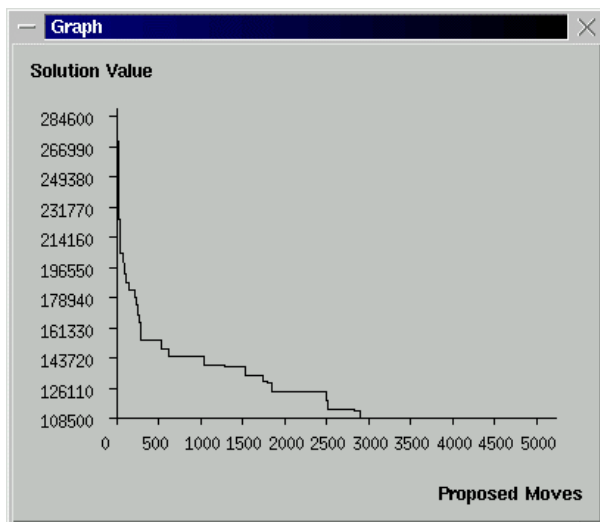


Figure 4: Solution Evolution Graph

Figure 4 shows the results obtained from a particular X-SAMSON search using the simple architecture presented in Figure 1. The initial solution was initialised with random values which had a fitness value of 284600. As the guided search progressed, the solution evolved towards better (lower scoring) architectures in an asymptotic fashion. After 395 seconds a solution score of 108500 had been obtained. This score was improved to 108000 after 863 seconds and later lowered further to 103000, but only after 1794 seconds. This demonstrates the computational complexity of the guided search and solution evaluation algorithms used within the search tool. The values quoted above were obtained using an AMD K6-2 400MHz processor running Linux. Another search using a Pentium 200 MHz processor had an initial architecture score of 233000, but took 517 seconds to evolve the solution to a score of 118000. It is obvious that computational power is a necessity with even the simplest of architecture models.

Our current target is to be able to analyse (a system based on) an industrial technology demonstrator, which is being used to demonstrate integration of flight and engine control functionality, on an IMA platform.

Integrity of the Heuristic and Future Aims

The Simulated Annealing search algorithm is non-deterministic and as such cannot be guaranteed to find an optimal solution. However, it can be set up to be innately pessimistic; that is if it indicates there is a solution there is one. The trade-off for this is that it may fail to find a solution that exists. Thus there is a safety-performance trade-off for the algorithm. However, it is possible to undertake extra analysis on the equivalent architectures and their accompanying reconfiguration tables after the algorithm has run to

show their validity. In practice, we expect to have to use this sort of additional off-line analysis to provide evidence that configurations really are “equivalent”, due to the difficulty of showing the soundness of the heuristics.

We plan to extend this work in a number of ways. For instance, we can investigate architectures under more extensive multiple failures. In this case we would need to extend our approach to consider degradation scenarios. That is, situations where we remove less vital functionality from the system. Three levels of importance would appear reasonable; essential, desirable and optional⁵. Restrictions on the set of admissible allocations can be introduced, for example preventing more than one replica of a process being executed on a processor in a particular physical zone of the aircraft.

There are many other issues which could be addressed, e.g. refining the model of failures to deal with safety more explicitly, or to investigate the utility of other heuristics such as genetic algorithms (Beasley et al. 1993). Our intention is to direct our approach to try to produce the simplest analysis framework which will scale to industrial problems, whilst giving credible and useful results.

CONCLUSIONS

The work outlined in this paper is, we believe, a novel approach to the problem of safety analysis and certification of dynamically reconfigurable IMA systems. By modelling and analysing the system off-line a static and deterministic table of legal reconfigurations in the presence of specific failures can be generated. Since these reconfigurations are known before runtime, each can be verified and certified. More importantly, by verifying the baseline system, and arguing that the rest are equivalent, it should be possible to certify a set of possible configurations at little more than the cost of demonstrating the acceptability of the baseline system.

Technically, the usefulness of this approach depends on the detail of the system model used and the criteria employed within the value (fitness) function used to score the model under analysis. External verification can be employed on the results to provide evidence for their validity. If the model detail and the fitness function criteria are sufficient, then we believe that this technique will provide a partial solution to a very difficult problem in the safety analysis and certification of the next generation of avionics systems.

Commercially, the usefulness of the approach is that it may offer the ability to certify IMA applications

⁵ Different segments of the industry use different terminology, but these concepts are quite widely used.

at a cost proportional to the scale of the application, not the number of legal configurations. However, in order to realise this benefit, it is necessary to get “buy in” from the certification authorities. We are presenting our preliminary results in the hope of getting feedback on the acceptability of the approach to the authorities, and to see how likely it is to satisfy its commercial as well as technical objectives.

REFERENCES

- Beasley, D., Bull, D., and R. Martin. (1993). “An Overview of Genetic Algorithms: Part 1, Fundamentals.” *University Computing*, 15(2), 58-69.
- Blackwell, N., Leinster-Evans, S., and Dawkins, S. K. (1999). “Developing Safety Cases for Integrated Flight Systems.” *IEEE Aerospace Conference*, 2408 Palm Avenue, Manhattan Beach, CA 90266 USA.
- Bradley, J., Fletcher, M., Miller, P., Moxon, P., and Wake, A. (1996). “Integrated Modular Avionics & Certification - An IMA Design Team's View.” *IEE Seminar: Certification of Ground/Air Systems*, Savoy Place, London, WC2R 0BL.
- Burns, A. (1995). “A Preemptive Priority-Based Scheduling: An Appropriate Engineering Approach.” *Advances in Real-Time Systems*, Prentice-Hall, 239.
- Chern. (1992). “On the Computational Complexity of Reliability Redundancy Allocation in a Series System.” *Operations Research Letters*, 11, 309-315.
- DoD. (1993). “Military Standard 882c: System Safety Program Requirements.”, US Department of Defence.
- Dowland, K. “Variants of Simulated Annealing for Practical Problem Solving.” *Adaptive Computing & Information Processing*, 115-133.
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). “Optimisation by Simulated Annealing.” *Science*, 220, 671-680.
- Mann, J. (1997). “SAMSON v1.6 User Manual.”, School of Information Systems, University of East Anglia.
- MoD. (1996a). “Defence Standard 00-55: Requirements for Safety Related Software in Defence

Equipment.”, UK Ministry of Defence.

- MoD. (1996b). “Defence Standard 00-56: Safety Management Requirements for Defence Systems.”, UK Ministry of Defence.
- Nicholson, M. (1998). “Selecting a Topology for Safety-Critical Real-Time Control Systems.”, University of York, York.
- Ribeiro, C., Treleavan, P., and Alippi, C. (1994). “Genetic Algorithm Programming Environments.” *Computer*, 27(6), 27-45.
- SAE. (1996). “ARP 4754: Certification Considerations for Highly-integrated or Complex Aircraft Systems.”, SAE.
- Sahner, R., Trivedi, K., and Puliafito, A. (1995). *Performance and Reliability Analysis of Computer Systems. An Example-based Approach using the SHARPE Software Package*, Kluwer Academic Press.

BIOGRAPHIES

Paul Hollow graduated from the University of York in 1997 with a degree in Computer Systems and Software Engineering (Meng). Since then he has worked as a Research Student in the Dependable Computing Systems Centre (DCSC) sponsored by BAE SYSTEMS at the University of York.

John McDermid is Professor of Software Engineering at the University of York. He directs the High Integrity Systems Engineering group at the University which is a recognised centre of excellence in safety critical computing for the UK Aerospace industry. He directs the Rolls-Royce University Technology Centre (UTC) in Systems and Software Engineering and the DCSC. He has published 6 books and over 200 papers.

Dr Mark Nicholson is a research and teaching fellow in the High Integrity Systems Engineering group at York. He graduated in 1986 and was awarded his DPhil in 1998. He teaches on the MSc course in Safety Critical Systems at the University of York and on-site safety courses for numerous industrial clients. His research interests relate to the use of quantitative methods in safety critical systems design and the certification of IMA. He is currently part of PAMELA, a European Union funded project, looking at the introduction of IMA into future commercial aircraft.