Seventh American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies
NPIC&HMIT 2010, Las Vegas, Nevada, November 7-11, 2010, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2010)

ASSESSMENT AND QUALIFICATION OF SMART SENSORS

Sofia Guerra[1], Peter Bishop[1,2], Robin Bloomfield[1,2] and Daniel Sheridan[1]
Adelard[1] CSR, City University[2]
10 Northampton Square, London EC1V 0HB, UK
{aslg,pgb,reb,djs}@adelard.com

ABSTRACT

This paper describes research work done on approaches to justifying smart instruments, and in particular, how some of this research has successfully been applied to the safety substantiation of such instruments. From a management perspective, we examine both the issues involved gaining access to information required for this justification and the necessity for a sustainable long-term approach for the justification of smart sensors that is acceptable to both suppliers and customers. From a technical perspective, we examine both overall safety justification approaches and specific techniques that can be used in the justification of the instruments' software. Our smart device assessment work covered both management and technical issues. Many of the approaches that were initially developed in research projects have now been applied in practice to smart devices that will be used in nuclear applications. We anticipate that further analysis techniques developed in our research programme will be deployed in future device assessments.

Key Words: Smart devices, safety assurance, justification of digital I&C instruments

1 INTRODUCTION

The nuclear industry is increasingly replacing analogue sensors with their digital "smart" counterparts. Smart sensors can achieve greater accuracy, better noise filtering together with in-built linearisation, and provide better on-line calibration and diagnostics features.

However, there are often difficulties with the safety justification of such instruments. Smart sensors are a specific form of COTS (commercial off-the-shelf) products, which are normally sold as a "black box" where there is no knowledge of the internal structure or their development process. Nevertheless, their safety justification, particularly for the more critical applications, might require knowledge of the internal structure and development process. The justification of sensors is made more difficult because the software constitutes a valuable intellectual investment, and the civil nuclear companies purchase sensors in small quantities.

Given the difficulty in obtaining replacement analogue sensors and the potential benefits of smart sensors, it is important that the nuclear industry develops an approach for justifying their use in safety and safety-related systems. This paper describes research done on approaches to justifying smart instruments, and in particular, how some of this work has successfully been applied to the safety substantiation of such instruments.

2 MANAGEMENT ISSUES

Over the years we have established relationships with several smart instrument manufacturers that have enabled us to obtain design and process information and, in most cases, the source code of their smart instruments. Establishing these relationships was a lengthy process spread over many months, involving several meetings. Once these were established there were fewer difficulties with our requests for further data on the devices or even the source code of other devices. The manufacturers supplied, to varying degrees, design documentation and additional data such as process and reliability data and certificates that could be used to support the safety justification of the devices.

This work started as a task in a research project to explore the feasibility of obtaining the source code and other information that could be used to justify smart instruments.

During our interactions with a range of manufacturers, some common long-term issues emerged:

- The suppliers expressed concerns about the effort and cost required for routine justifications of smart sensors—the nuclear industry is a small market compared to other sectors, and the expense might be excessive relative to the potential sales.
- Suppliers are generally in favour of an “assurance package” of additional information that is paid for by the customer.

Some aspects of these long-term issues were addressed by the development of a tool supported questionnaire (that we have recently re-implemented for the nuclear industry): *Emphasis* [9]. This reflects a UK nuclear interpretation of IEC 61508 for smart instruments. This tool has the potential to be the basis of a package that would be used in the qualification of a smart instrument for a number of applications.

3 JUSTIFICATION OF SMART INSTRUMENTS: THE UK CONTEXT

The UK has a specific approach to how it assesses and licences command, control and protection systems. Despite the internationalisation of the supply chain and effective collaboration with international agencies (IAEA, OECD), standards committees (IEC), working groups (NRWG) and projects to encourage harmonisation (such as Cemsis [4]) there are still significant differences between the UK and other countries.

The UK Health and Safety Executive (HSE) Safety Assessment Principles (SAPs) [8] are the primary principles that define the overall approach to be followed for nuclear installations in the UK. The SAPs have been revised in the past few years and have been brought into line with IAEA guidance.

The SAPs have the following clauses on computer-based safety systems.

> Where the system reliability is significantly dependent upon the performance of computer software, the establishment of and compliance with appropriate standards and practices throughout the software development life-cycle should be made, commensurate with the level of reliability required, by a demonstration of ‘production excellence’ and ‘confidence-building’ measures.
>
> ‘Production excellence’ requires a demonstration of excellence in all aspects of production, covering initial specification through to the finally commissioned system, comprising the following elements:
>
> a) Thorough application of technical design practice consistent with current accepted standards for the development of software for computer-based safety systems.
>
> b) Implementation of an adequate quality assurance programme and plan in accordance with appropriate quality assurance standards.
>
> c) Application of a comprehensive testing programme formulated to check every system function.

Independent 'confidence-building' should provide an independent and thorough assessment of a safety system's fitness for purpose. This comprises the following elements:

a) Complete and preferably diverse checking of the finally validated production software by a team that is independent of the systems suppliers, including:

• independent product checking providing a searching analysis of the product;

• independent checking of the design and production process, including activities needed to confirm the realisation of the design intention;

b) Independent assessment of the test programme, covering the full scope of test activities.

Should weaknesses be identified in the production process, compensating measures should be applied to address these. The type of compensating measures will depend on, and should be targeted at, the specific weaknesses found.

The justification approach used for smart instrument needs to be consistent with these clauses to be acceptable for safety-related systems in the UK nuclear industry.

4 JUSTIFICATION APPROACH

There are different strategies that can be deployed in the safety justification of smart sensors. The three main approaches can be characterised as a "triangle" of

- the use of accepted standards and guidelines
- justification via a set of claims/goals about the system's safety behaviour
- an investigation of known potential vulnerabilities of the system

This is illustrated in Figure 1.

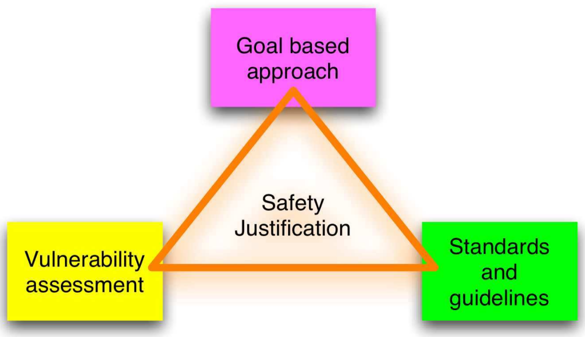

Figure 1. The Safety justification triangle

The first approach is based on demonstrating compliance to a known safety standard. This is a common strategy. For example, the UK C&I Nuclear Industry Forum (CINIF) sponsored the development of the *Emphasis* tool that supports an initial assessment of compliance with IEC 61508.

The second approach is goal-based – where specific safety goals for the systems are supported by arguments and evidence at progressively more detailed levels. This would typically be implemented using Claims-Argument-Evidence (CAE [1]) or goals-structuring notation (GSN [14]) notations.

The final approach is a vulnerability-based argument, where it is demonstrated that potential vulnerabilities within a system do not constitute a problem. This is essentially a ‘bottom-up’ approach as opposed to the ‘top-down’ approach used in goal-based methods.

These approaches are not mutually exclusive, and a combination can be used to support a safety justification, especially where the system consists of both off-the-shelf components and application-specific elements. More specifically, the requirement for “production excellence” in the HSE SAPs can be met by demonstrating compliance to an acceptable standard, while the requirement for “independent confidence building measures” can be addressed by a combination of goal-based assessments (e.g. demonstration of accuracy, reliability, etc.) combined with assessment of potential vulnerabilities in the smart device implementation. We have undertaken research on these alternative justification approaches and applied this research in the justification of these smart devices for actual nuclear applications.

4.1 Standards Compliance

For the standards compliance, we have assessed the development process of a number of smart instruments. The assessment was support by the *Emphasis* standards compliance tool. We have also re-engineered the tool so that it can be used either standalone or via a web-based server, where it can be accessed by a standard web browser. While the current focus is compliance against IEC 61508 [10], the tool is database driven, so different sets of assessment questions could be used to assess compliance to different standards. The tool allows the smart device manufacturer to provide answers to the questions and upload relevant supporting documents. A nominated independent assessor can review the supplier responses and assess their adequacy. Typically, this assessment is followed by an audit of the supplier processes, which will confirm or reject the answers given by the supplier. The tool provides an overview of the assessments, summarising the number of questions answered and the number of questions that have been accepted by the assessor.

Usually, an assessment would identify areas where there are gaps and weaknesses in the supplier process. In these cases the tool assessment is supplemented by other activities such as tests that may exercise some specific behaviour of the instrument not considered by the supplier’s verification activities.

4.2 Goal-based Assessment

A goal-based assessment [2] can be used to build confidence that a smart device “does what it says on the tin”. It may also be used to focus the review of the verification activities performed by the manufacturer and ensure that all relevant behavioural attributes have been demonstrated, e.g., accuracy or time response.

We have been developing a goal-based approach for smart instruments that is presented in the form of set of claims in a safety case, namely that

- the specification of the smart device is adequate
- the smart device behaves according to the specification

The key advantage of a goal-based approach is that there is considerable flexibility in how the top-level claims are demonstrated since different types of arguments and evidence can be used. For example, the device might not have much product development evidence, and hence a standard compliance case may be difficult to justify, but it may have extensive field experience and records of field-reported faults that demonstrate reliable operation. So, this might be used as an alternative means of demonstrating adequate product quality and compliance with specified behaviour.

Alternatively, evidence could be explicitly generated to demonstrate goal-based claims about compliance with specified device behaviour such as functionality, time response or robustness to abnormal inputs. This evidence could be generated using a range of assessment techniques including static analysis, black-box testing and field experience (which are considered in more depth in Section 5).

At the moment we are developing an approach that combines *Emphasis* with a goal-based approach. This approach benefits from being more flexible than a strict rule-based approach, giving rationalisation and justification for any gaps in the assessment and how compensation for these gaps has been achieved. Although this approach has not been used to justify instruments for real applications, we expect to conduct pilot studies of this combined approach in the near future.

4.3 Assessment of Potential Vulnerabilities

There are a range of potential vulnerabilities in smart device software that could affect its behaviour. Typical examples might be buffer overflow, numeric overflow or use of non-initialised data. A vulnerability assessment for a smart device would normally require access to the source code and require some form of static analysis of the code to identify specific types of vulnerability.

This type of assessment can increase confidence if no vulnerabilities are found, but it is difficult to argue that all possible types of vulnerability have been considered. In practice, for smart devices with a low safety integrity target, the software is examined for a specific set of vulnerability types (e.g., where the analysis can be supported by tools). So a vulnerability assessment can be viewed as a means of *challenging* a claim that a smart device is adequately safe, i.e., it can help to refute the validity of a smart device justification (because it does the wrong thing) but it cannot be used as the only support for a claim that a smart device is safe (i.e., it always does the right thing).

5 ANALYSIS TECHNIQUES

Over the years, the smart sensor manufacturers that we have built relationships with have provided us with source code for several sensors. This code has been the subject of a series of research projects designed to identify and evaluate potential ways of analysing smart sensor code in order to build confidence in its correctness. Both goal-based and vulnerabilities-based assessment approaches have been used. We have explored techniques which provide general confidence in code quality and correctness as well as techniques aimed at identifying specific code flaws. Sometimes these two approaches are very closely related.

5.1 Smart Sensor Code

The code that we have examined has a number of distinctive features:

- assemby language was used in early devices, but C code is found in most modern smart devices
- the code size is small – tens of thousands of lines of code, although this may increase significantly if the instrument includes fieldbus or other type of communication protocols
- there is no distinct operating system – the code manages the hardware directly
- interrupts are used for device input outputs and timing (so there are concurrent code threads)

These features are not surprising for small, embedded systems, but help us to scope the types of analysis that are suitable. They also place smart sensors at a particular “sweet spot” as far as analysis research is concerned: the systems are simple enough to be amenable to a wide variety of techniques, while still providing real, practical examples where the outcome of the analysis has value. In the following sections, we will describe the different forms of analysis that have been used to justify smart device software.

5.2 Use of Integrity Static Analysis to Detect Potential Vulnerabilities

Integrity static analysis [16] is the use of a range of static analysis approaches to identify specific classes of vulnerabilities. Typically the tools used to perform these analyses can generate “false positives”. To address the problem of false positives we adopt a sentencing approach to the static analysis results as follows:

- Provisional sentencing – based on inspection of the code associated with the finding, and classifying the finding according to its perceived impact.
- Domain expert sentencing – any non-trivial findings are discussed with the manufacturer to determine which can be resolved and which are genuine.
- Final sentencing – the remaining findings are sentenced according to impact and likelihood.

The specific techniques applied in integrity static analysis are discussed in more detail in the sections below. These techniques are far easier to apply to C code than to assembly code as there are a wide range of tools that support these analyses. For assembly code, we have explored an analysis route that involves translating assembler to C. This allows us to analyse some aspects of the structure of the code using tools aimed at C. However, a full translation can be difficult to implement since we lack information about data types and structures and control flow in assembly code may violate the constraints imposed on C programs. For example, we have found different assembler subroutines that jump to a shared body of common assembly code. While some automated code analyses were performed on assembler code, most analyses had to be implemented by manual review.

5.2.1 Unsafe language constructs

The use of C in smart sensors makes it is easy to use ill-defined constructs or to write code that is poorly structured. For higher-integrity devices, we have seen a trend towards language subsetting (e.g., Misra-C) which removes many of these constructs. However, we have found that applying C coding style analysis to code not deliberately written with a subset in mind can still produce useful results. Our approach has been to use a tool to automate potentially unsafe uses of C constructs and then filter the results according to their practical impact discarding many issues which are considered bad style but can easily be assessed as not affecting the behaviour of the device.

5.2.2 Control and data flow analysis

Here we look for control flow problems like unreachable code and poorly structured code (like using *goto*). We also look for potential data flow errors like reading of uninitialised variables or writing a variable without a subsequent read.

5.2.3 Runtime error analysis

This type of assessment covers vulnerabilities such as array bound overflows, numeric overflow, divide by zero and stack overflow (in cases where only limited stack space is available).While some cases can be detected with simple tools (like possible cases of divide by zero), automation of such analyses requires more specialist tools (such as *PolySpace* [11]) that use abstract interpretation to determine the range of potential values of a variable at run-time. This type of analysis can be resource intensive limitting the size of program code that can be analysed. We have applied this analysis to software performing a critical control application on a nuclear plant (but not to a smart device).

5.2.4 Concurrency analysis

Smart sensors are typically structured as simple interrupt-driven devices, without a true multitasking operating system. Nevertheless, some aspects of concurrency are particularly important: variables may be shared between continuously-executing code and the interrupt handlers, and are subject to a subset of the usual race conditions experienced by concurrent code. This is illustrated in Figure 2 below, where the notification of a safety problem within the interrupt code is overwritten when the mainline code notifies a different, more trivial problem at the same time.

Main code thread	Interrupt handler thread
temp = **err_status**;	
/* interrupt occurs ---> */	void interrupt_handler (void) {
	int temp2;
	temp2 = **err_status**;
	err_status = temp2 & safety_error;
	Return;
/* return from interrupt <--- */	}
err_status = temp & trivial_error;	

Figure 2. Example interrupt hazard.

The peculiar nature of interrupt-driven code means that generic tools and techniques for analysing concurrent code are not easy to apply. In addition, as we are analysing existing code rather than developing new code, we cannot simplify the concurrency analysis by imposing specific rules on the design of the software—it must be analysed “as is”.

Our initial research in this area was based around modelling the execution sequences of the code, as represented by its flowgraph, and combining this with a model of the interrupt behaviour of the CPU in question (for example, whether interrupt handlers can themselves be interrupted). While generating very large models, this approach enabled us to identify hazards such as those shown in Figure 2. We have now developed a simpler approach which identifies the same hazards based on enumerating the variables shared between the main code and interrupt handler, and types of reads and writes that they are subjected to.

Another analysis we perform is the identification of code used on multiple threads. A typical case is where the mainline code and the interrupt code call a common subroutine. Execution of common code in different threads in C is not a problem as the code should be re-entrant, but if the common routine manipulates global variables, the data update can be interrupted partway through, so the update can be corrupted.

We plan to research other types of concurrency “glitch”. In particular, smart devices often communicate with input-output systems using a sequence of commands sent to a device register (or set of device registers). It is important that other threads do not use the device register set during this sequence. The analysis would identify the code where the register interaction sequences are performed and check they cannot be affected by code executing in other threads.

5.3 Formal Proof

Formal verification techniques such as proof of correctness have traditionally only been applied to high criticality systems, due to their high cost. However, in many systems, particularly embedded systems such as smart sensors, only a small fraction of the code is performing high criticality functions. Our approach, called "focused proof", makes formal proof applicable at all safety integrity levels. A combination of techniques is used to achieve this, as described below.

- Use of modern proof tools. For example the Frama-C code analysis framework [6] with the Jessie verification condition generator [7] is able to deal directly with industrial grade C code, and interface with modern decision procedures such as Yices [15] and CVC3 [5].
- Restricting attention to critical code. A code review identifies the regions of the code which are directly responsible for the process variable. This code may be as little as 20% of the codebase, and is typically comparatively simple. The code typically implements simple scaling and other transformation functions on the process variable. This means it can be proved correct quite easily.
- Independence analysis. The code which deals with user interaction, housekeeping, etc., is audited to identify points of interaction with the critical code. We can apply a variety of techniques to show independence, such as code slicing to establish that the value of a configuration variable shown on screen is the same as the value used by the critical code.

As an alternative to the independence analysis, we could also use testing to exercise the parts of the code not covered by the formal analysis. This may be rolled into an existing test programme already planned for the device.

For some parts of the code, the approach described above may not be sufficient. For example, in one example that we encountered, a sequence of large switch statement was used to determine the device behaviour in any of several dozen input sensor types (selected by a configuration parameter). Determining the actual behaviour of the device proved difficult. In this case, we used partial evaluation techniques, which specialise a program according to a particular set of configuration parameter values. By deriving a version of the program specialised to a single input sensor type (like a particular type of thermocouple), the computation became very clear and was easy to analyse. This strategy can be effective when only a limited number of the device configuration options are intended to be used in actual nuclear applications.

5.4 Black-box Testing

The static analysis approaches described in the previous sections require access to the source code. In practice, it may not always be possible to obtain the source from a supplier. As an alternative means of assessment, we are researching the options for black-box testing. In principle this can be used to evaluate all externally observable aspects of behaviour found in the smart device specification, including:

- functional behaviour (in different configurations)
- accuracy
- response time
- robustness (response to external failure conditions like power interruptions and broken connections)
- failure integrity (ensuring internal failures are detectable)

In practice, some of these attributes are easier to assess than others. For example, assessment of response to internal failures will require a deliberate introduction of a failure condition in the device which could result in permanent damage. To date we have focused on the black-box assessment of functionality and time response.

5.4.1 Functional testing

For the functional testing we have developed the three stage approach shown in Figure 3.

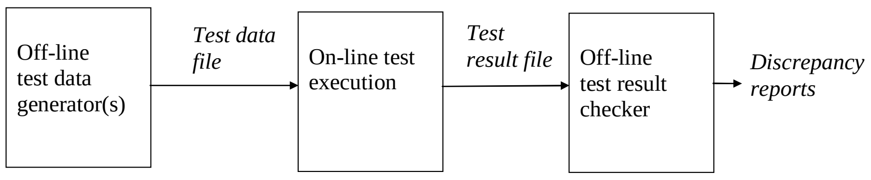

Figure 3. Black-box test phases

We use off-line test generation and checking because this gives us additional flexibility. If there are errors in the checker, the test result file is still valid, so we only need to fix the checker. In addition, we can use a range of different test generation strategies without affecting the design of the online test execution system or the off-line result checker.

A number of test generation strategies have been evaluated, including:

- Test data generated from a formal model of behaviour derived from the smart device specification. The language SAL was used to define the model and the *sal-atg* tool was used to generate test data to cover the different states of the model.
- Statistical tests. This is an approximation of plant transients where the input increases to some limit with random fluctuations. The test results can be used to support claims made in a goal-based assessment about correct functional behaviour to some specified level of reliability. For example, 4600 simulated transients without failure can support a claim of 10^{-3} failures per demand transient to 99% confidence.
- Random tests. These are designed to maximise “stress” on the smart device by maximising changes of state which might reveal obscure software defects.

Some examples of these types of test data are shown in Fig. 4 below.

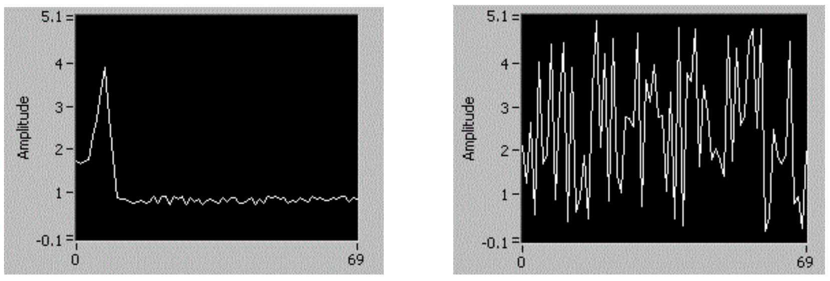

Figure 4. Example plant transient and random test data

We found that the checking of the test results is complicated by inherent non-determinism within the smart device. This arises because we do not know precisely *when* the smart device will read to the test input or precisely *what* value the smart device will read in. So if the smart device reads a test value close to a decision point, multiple responses would be valid given the specified accuracy and response time. We have examined ways of dealing with this (see [3]).

In addition to automated testing, we have looked at manual testing, primarily for testing the user interfaces used to configure the device. We use a negative testing approach where we attempt to enter invalid inputs such as:

- The wrong data type (e.g., text rather than numeric, fraction rather than integer)
- The right type but the wrong value (too large, too small, e.g. use zero or a negative value)

- Entering correct values in an abnormal order or terminating configuration before all values are completed.

This strategy has proved to be quite effective in revealing defects in the configuration interface software (particularly in remote configuration software that modifies the smart device via a communication link).

5.4.2 Time response testing

While response times are normally included in smart device specifications, the timing characteristics can be affected by undocumented features of the smart device such as

- the input sampling rate
- smoothing or filtering algorithms applied to the input data sequence

If the smart device retransmits the input value as a smart device output, there are some relatively simple ways of establishing the sample rate. Figure 5 shows the smart device response to a triangular input sequence produced by a signal generator.

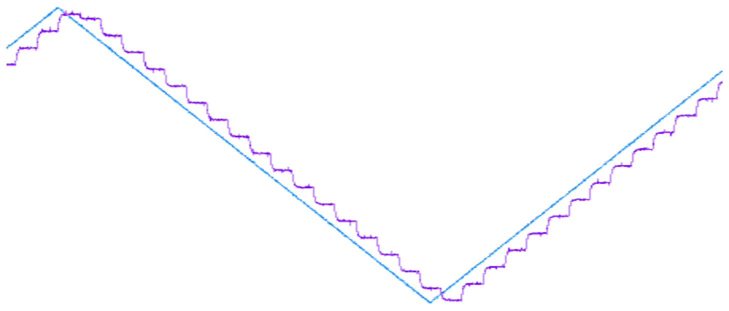

Figure 5. Response to a triangular input signal

The response is a sequence of “stair steps” and the time separation between steps indicates the sample rate. We can also see that there is some lag in the retransmitted value as it is not a sharp step – it takes a while for the transmitted output value to settle. The output response can become more complex when input smoothing is enabled, but it is possible to infer the smoothing algorithm from the characteristic response to a define input transient.

5.5 Analysis of Field Experience

Field experience can be a useful means of independent confidence building or of compensation for gaps in the excellence of production. Operation in the field could be viewed as an extended form of beta testing, which can reveal any residual software defects. This would normally require the cooperation of the smart device manufacturer to supply data on

- the number of units sold (which may include variants of a basic design)
- defect reports detailing when the defect was reported and the nature of the defect
- evidence that defects have been fixed in later software revisions
- information about the processes used for collecting failure reports from users and fixing the defects

This data needs to be assessed to determine if the feedback reporting mechanism is adequate and if sufficient operating time has been gained to ensure that most defects are found and reported.

Given an acceptable data set, typical analyses include:

- Review of the defects to check that none would have seriously impaired its specified functionality. This provides evidence for the quality of the smart device development processes which could supplement a standards compliance assessment (especially where the compliance evidence is weak).
- Looking for evidence of reliability growth, i.e., new defect reports decrease as smart device usage increases.

We have analysed smart devices with operating experience of up to 10 000 device years where the field evidence would support claims of a software MTTF of 100 years. We must however bear in mind that under-reporting by the smart device users could lead to over-optimistic estimates of the MTTF.

6 CURRENT WORK AND FUTURE APPLICATIONS

The research we have undertaken has branched into a number of projects on several aspects of the justification of smart instruments, which consider both justification approaches and analysis techniques. The research will continue the development of analysis techniques suitable for the justification of smarts instruments that are applicable at different integrity levels, including

- static analysis, including an extended form of concurrency analysis which take into account concurrently operating input-output hardware, and methods for estimating worst case time response
- black-box testing techniques, especially for assessing time-dependent response, robustness to abnormal input, and failure integrity (where the failure is external detectable)
- identifying sets of techniques that can be deployed to justify smart instruments at different levels of integrity targets and with different types of available evidence

In addition to our continuing research work, we have been involved in the justification of a number of smart sensors for the nuclear industry in the UK. These justifications have been done for specific nuclear applications and as part of a pre-qualification programme of smart instruments.

Several of the approaches we developed in our research programme (primarily static analysis and analysis of field experience) have been successfully applied in these justifications. In future justifications, we intend to use black-box testing as an additional confidence building measure. We have also integrated our assessment approach with the overall regulatory framework provided by the UK Safety Assessment Principles. We believe that this work provides a significant technical basis for sensor (and other device) assessments as well as reducing the resource and project uncertainties associated with using such devices.

7 CONCLUSIONS

Our smart device assessment work covered both management and technical issues.

- From a management perspective, we examined the issues involved with interacting with the suppliers to gain the information needed for the justification. We also addressed the need for a sustainable long-term approach for the justification of smart sensors that is acceptable to both suppliers and customers.

- From a technical perspective, we considered both overall safety justification approaches and specific techniques that could be used in the justification of the instruments' software.

Many of the approaches that were initially developed in research projects have now been applied in practice to smart device that will be used in nuclear applications. We anticipate that further analysis techniques developed in our research programme will potentially be deployed in future device assessments.

8 ACKNOWLEDGMENTS

The work presented in this paper was partly funded by the C&I Nuclear Industry Forum in the UK.

9 REFERENCES

1. P. G. Bishop and R.E. Bloomfield, "The SHIP Safety Case - A Combination of System and Software Methods", in *SRSS95, Proc. 14th IFAC Conf. on Safety and Reliability of Software-based Systems,* Bruges, Belgium, 12-15 September 1995.
2. P. Bishop, R. Bloomfield and S. Guerra, "The future of goal-based assurance cases," *Proceedings of Workshop on Assurance Cases*. Supplemental Volume of the 2004 International Conference on Dependable Systems and Networks, pp. 390-395, Florence, Italy, June 2004.
3. P. G. Bishop, L. Cyra, "Overcoming Non-determinism in Testing Smart Devices: A Case Study", in *Proceedings of the 29th International Conference on Computer Safety, Reliability and Security (SAFECOMP 2010)*, Vienna, Austria, 14-17 Sept 2010, pp.327-250. (2010).
4. CEMSIS - Cost-effective modernisation of systems important to safety. A Nuclear Energy research project under the European Union FP5 programme http://cemsis.org.
5. "CVC3 Home Page", http://www.cs.nyu.edu/acsys/cvc3/.
6. "Frama-C, Software analysers", http://frama-c.cea.fr/.
7. "Frama-C, Jessie plug-in", http://frama-c.cea.fr/jessie.html.
8. HSE, "Safety Assessment Principles for Nuclear Facilities", http://www.hse.gov.uk/nuclear/saps/index.htm.
9. R. Stockham, "Emphasis on Safety", in *IET Magazine*, Issue 02 2009.
10. IEC, "Functional Safety: Safety-related Systems", IEC 61508, (Parts 1 to 7), *International Electrotechnical Commission*, 1995.
11. "Mathworks PolySpace", http://www.mathworks.com/products/polyspace/index.html.
12. MISRA-C:2004, *Guidelines for the use of the C language in critical systems*, ISBN 095241564X.
13. "Symbolic Analysis Laboratory (SAL)", http://sal.csl.sri.com/.
14. S. P. Wilson, T. P. Kelly, J. A. McDermid, "Safety Case Development: Current Practice, Future Prospects", in *Proceedings of 12th Annual CSR Workshop*, Bruges, Belgium 13 Sept 1995 (Springer-Verlag).
15. "Yices: An SMT Solver", http://yices.csl.sri.com/.
16. P.G. Bishop, R.E. Bloomfield, T.P. Clement, A.S.L. Guerra and C.C.M. Jones, "Integrity Static Analysis of COTS/SOUP", in *Proceedings SAFECOMP 2003*, pp. 63-76, 21-25 Sep, Edinburgh, UK, 2003, (c) Springer Verlag.