Justification of Smart Sensors for Nuclear Applications

Peter Bishop[1,2], Robin Bloomfield[1,2], Sofia Guerra[2], and Kostas Tourlas[2]

[1] CSR, City University
[2] Adelard, Drysdale Building, 10 Northampton Square, London,
EV1V 0HB, UK
{pgb, reb, aslg, kt}@adelard.com

Abstract. This paper describes the results of a research study sponsored by the UK nuclear industry into methods of justifying smart sensors. Smart sensors are increasingly being used in the nuclear industry; they have potential benefits such as greater accuracy and better noise filtering, and in many cases their analogue counterparts are no longer manufactured. However, smart sensors (as it is the case for most COTS) are sold as black boxes despite the fact that their safety justification might require knowledge of their internal structure and development process. The study covered both management aspects of interacting with manufacturers to obtain the information needed, and the technical aspects of designing an appropriate safety justification approach and assessing feasibility of a range of technical analyses. The analyses performed include the methods we presented at Safecomp 2002 and 2003.

1 Introduction

Sensors for nuclear applications have been relatively simple analogue devices with known performance properties and known failure characteristics. However, the sensor industry is increasingly using microprocessor-based "smart sensors". Smart sensors can achieve greater accuracy, better noise filtering together with in-built linearisation, and provide better on-line calibration and diagnostics features. So, given the difficulty in obtaining replacement analogue sensors, and the potential benefits of smart sensors, it is important that the nuclear industry develops a suitable approach for justifying the use of smart sensors in systems important to safety (SIS).

Smart sensors are a specific form of COTS (commercial off-the-shelf) product. COTS products are normally sold as a "black box" where there is no knowledge of the internal structure. However, their safety justification might require knowledge of the internal structure and development process. The justification of sensors is an increasing problem because the software constitutes a valuable intellectual investment, and the civil nuclear companies purchase sensors in small quantities.

This paper presents the results of a research study sponsored by the UK nuclear industry into methods for justifying smart sensors. The project has covered both management and technical issues.

- From a management perspective, we examined the issues involved in 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 need an assessment approach that is both proportionate and feasible to apply and is commensurate with the SIL of the intended application(s). The approach should also be related to existing assurance requirements for computer-based systems in nuclear application (e.g. HSE SAPs [1] and the British Energy PES Guidelines [2]) and should address the concerns related to the black box assessment of smart sensors (and other COTS products).

The paper is structured as follows. In Section 2 we describe the interactions with the manufacturers that took place in the scope of this project, as well as the main issues identified during these interactions. Section 3 summarises three different viewpoints of the safety justification of smart sensors, including a goal-based approach that is expanded in Section 4 and vulnerability assessment summarised in Section 5. Section 6 describes the analyses performed on one of the smart devices obtained, while Section 7 relates these analyses with the three approaches from Section 3. The conclusions are presented in Section 8.

2 Relationships with Smart Sensor Manufacturers

2.1 Obtaining Software, Supporting Data and Company Culture

Obtaining the software was a lengthy process spread over many months, involving several contacts, phone calls and negotiation of conditions for non-disclosure of the results. However, once the relationship was established, there were fewer difficulties in providing further data on the devices or even the software of other devices.

As a result of these negotiations, we successfully obtained smart sensor software from two manufacturers, and we were offered the possibility of obtaining another example by a third manufacturer. In addition, 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.

It must be borne in mind, however, that the software was provided on the understanding that it was a part of a research project undertaken by a specialist third party. Whether a similar level of access would be provided to end-users on a routine basis remains an open question.

The nature of the relationships with the smart sensor manufacturers was quite distinct, probably as a result of the key markets they target and the management structure and particular characteristics of the companies. It is clear that smart sensor suppliers can have different company cultures, and this can have a major impact on the feasibility of gaining access to the source code and subsequently a successful assessment. Some pertinent questions to assess the likely success of the interaction might be:

1. How does the organisation deal with this type of interaction? Are there precedents in the past?
2. Who is the gatekeeper and what is their role in the organisation? (a "gatekeeper" is an official point of contact in the company who decides whether to allow requests to pass to higher management):

- Do they have access to design authority?
- What are the lines of communication?
- How frequently and with whom are meetings available to clarify issues?

3. How difficult would access to management be?
4. To what extent does the organisation see the relationship as benefiting them?
   - What are the main market sectors?
   - Is nuclear industry a main market, a niche or a distraction?
   - Are the benefits of engagement seen only in terms of future sales?
   - Does the organisation see benefits of interaction from a technical or process improvement point of view?
5. What is the attitude to confidentiality?
   - Is a standard non-disclosure agreement sufficient?
   - Is the confidentiality agreement with the assessor company or with named people from the company?

2.2 Long Term Issues

Despite the differences between the supplier approaches, some common long-term issues emerged:

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

The big issue is what this package should contain and whether it will be acceptable to a wide range of customers (e.g. the nuclear industry or beyond). There is a UK initiative to define a framework of IEC 61508-conformant documentation about the development processes for a device; this has the potential to form part of an assurance package. The SIREP sensor assessment could also be used in support of the functional and hardware performance of the sensor. Further work would be needed to describe the full content of such a package. The main area of weakness is the “black box” nature of such evidence, and greater confidence could be obtained if there was some knowledge of the internal design and implementation of the device (e.g. “grey box” information like design documents, or white box information like source code).

Independent certification is used to provide “black box” evidence for performance (e.g. accuracy) and environmental withstand (e.g. electrical isolation, temperature). Evidence also exists for hardware reliability and safe failure fraction (e.g. using hardware reliability models and FMEDA). But independent assessment of software (like a TUV assessment for compliance to IEC 61508) is not routinely performed. This is thought to be quite costly especially if it has to be updated with each revision.

It is clear there is a perceived tension between the need for suppliers to maintain confidentiality and the user’s requirement to provide sufficient evidence to demonstrate safety. In principle the licensee should always be able to check the evidence and the details of the analysis performed. This might not be feasible for a compliance certification approach where a certificate is provided but the rationale and detailed evidence is not. By contrast “black box” system certification against test standards seems less of a problem as the specific tests undertaken are clearly defined. Perhaps third party software evidence would be more acceptable if a common set of software analyses and tests were defined that could be subjected to external audit.

3 Safety Justification Approaches

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:

- Justification via a set of claims about the system’s safety behaviour.
- The use of accepted standards and guidelines.
- An investigation of known potential vulnerabilities of the system.

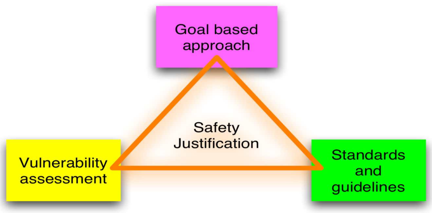

Fig. 1. Safety case approaches

The first approach is goal-based—where specific safety goals for the systems are supported by arguments and evidence at progressively more detailed levels. The second approach is based on demonstrating compliance to a known safety standard. 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 (OTS) components and application-specific elements.

In the past, safety justifications tended to be implicit and standards-based—compliance to accepted practice was deemed to imply adequate safety. This approach works well in stable environments where best practice was supported by extensive experience, but with fast moving technologies, a more explicit goal-based approach has been advocated, which can accommodate change and alternative strategies to achieve the same goal.

4 Goal-Based Safety Justification

4.1 Goal-Based Justification of COTS Products

Goal-based approaches are often used in safety justifications ([3], [4], [5], [6]). This is a flexible approach as it focuses directly on the safety requirements for the sensor and can be related to a range of different safety standards by identifying how the standards' requirements support the various claims.

HSE guidance on COTS/SOUP ([7], [8]) recommends that the goals for a computer-based system are related to factors that directly affect safety, e.g.:

- functional behaviour
- accuracy
- reliability and availability
- fail-safe behaviour
- time response

A similar set of attributes could be identified for a smart sensor component of a system. Moreover, we have to recognise that a smart sensor justification is part of a larger safety justification for a "System Important to Safety" (SIS).

1. The justification of the component can be used to show that the component "does what it says on the tin". This is application independent.
2. The SIS safety justification has to show that the component is suitable for the application context and satisfies any constraints.

For example, there could be a component claim that a smart sensor is accurate to 10^{-3} and an application-level claim of 10^{-2} for the accuracy of some computation involving the combination of several measured values. Alternatively, the sensor might explicitly include functionality that is not needed (e.g. support for different types of resistance thermometer). In this case we need to show in the SIS safety justification that the unwanted functions are not activated. In addition there may well be functionality that is not declared as part of the product.

The extent of evidence to justify specific sensor properties will vary with the required integrity level, and the type of evidence required may also need to change with the integrity. At lower integrity levels there is greater emphasis on "black box" evidence (externally observed behaviour) and evidence of development process. Yet, it is necessary to look "inside the box" for greater assurance and higher integrity applications. Indeed the public consultation on the HSE study showed a consensus that the assessment of critical SOUP should include white box assessment.

4.2 Key Goals

In a goal based approach we identify the key attributes required for a smart sensor, and then seek arguments and evidence to show these goals are met. Clearly many properties like environmental limits (temperature, humidity, supply voltage) and resistance to interference (RFI, EMI, mains noise) are only dependent on the hardware and can be justified in the same way as conventional hardware. However, other properties will depend on a combination of hardware and software and may therefore require different or additional supporting evidence to address software concerns.

We focused on the properties of the system that involve software. Based on the specifications of the two suppliers and more general consideration of smart sensor requirements, the following set of goals was identified that are likely to involve software in the smart sensor, where the values used correspond to those of the temperature transmitter specification. This could be extended to include functional capabilities of the device.

Table 1. Smart sensors properties for goal-based approach

Ref	Sensor property
1	Output conversion accuracy better than 0.03% under stated conditions
2	Sample rate < 128 ms, 10%-90% step response < 256 ms
3	Safe failure fraction >0.72
4	MTTF > 2.5 years (average)
5	Configuration and calibration errors are minimised
6	Smart sensor behaviour is predictable
7	Smart sensor properties 1-6 can be maintained for next 10 years

Note these goals are application independent, i.e. defined by the supplier. The numerical values will vary with the actual sensor, but these properties are directly relevant to the safety justification of the SIS as a whole. The system implementor has to define higher-level system goals (e.g. timeliness and accuracy of the overall system) and demonstrate that the properties of the sensor support the SIS safety justification.

5 Vulnerability Assessment

The vulnerability assessment focused on a number of concerns with a purely black-box based approach. Discussion of these concerns is not included in the paper, but included adequacy of functional testing, software security vulnerabilities and malicious code among others. In subsequent analysis we expanded the final category of "non-predictability" to consider specific sources of non-predictability, namely:

- concurrent interaction problems (non-atomic updates, deadlocks, etc.)
- non-initialisation of data
- data overflow
- variable time response (data dependent timing, infinite loops, etc.)

The black box concerns identified above could result in unexpected behaviour even if the equipment appears to conform to specification when functional testing of the "black box" is performed. Such concerns can be used as a checklist to determine whether the risks of potential vulnerabilities have been addressed (e.g. using additional grey-box or white-box evidence). The fundamental limitation of black-box analysis is the extent to which the testing/field experience profile used mirrors the actual use in the new application.

6 Analysis of a Smart Device

In Section 3 we described different aspects of a safety justification of a smart sensor, but no supporting evidence was presented. In this section we describe the technical analyses performed in the project concerning a specific sensor product: a temperature transmitter. The aim of this work was to assess the feasibility of a variety of techniques and analyses to support the safety justification of the temperature sensor.

The microprocessor is programmed in assembly language. The software comprises

- 6000 lines of assembly code (including comments)
- 7 kilobytes of binary code

The source code has been subjected to a range of assessments to support its safety justification, namely:

- Code criticality analysis. Identification of the code essential to operation and obsolescent code.
- Code structure analysis. Identification of concurrent program threads and checks for safe exchange of data between threads and absence of deadlocks.
- Code integrity assessment. Check for defective code constructs (e.g. array bound overflow, divided by zero, dead code, stack overflow).
- Redundant code analysis. To identify if there is any unexpected code.
- Failure integrity analysis. Assessment of the failure integrity features to check whether failures result in a safe state.
- Predictable execution assessment. Assessment of whether the ordering of software functions is well defined, of the input-output conversion accuracy and whether the execution time has a predictable upper execution time bound.
- The development of a strategy for statistical testing (not described in this paper).

The feasibility of performing such analyses depends on the code size, structure, and programming language. There is more tool support for high level languages (like C) for performing analyses. However, we have made use of assembler level software simulators to perform direct testing of internal functions within the software and to evaluate the code coverage and code execution times.

6.1 Code Criticality Analysis

We performed software criticality analysis [9] to assess the importance to safety of various components within the software. This showed that the code is separated into the main conversion function and a calibration function. The calibration code can only modify the calibration parameters used by the main code, and hence could be viewed as less critical as it is not executed in normal operation. Most of the analysis effort (such as the timing analysis) was focused on the main conversion function.

6.2 Structural Analysis and Concurrency Analysis

In order to perform evaluations of the sensor software it was first necessary to understand the overall software architecture. To gain an understanding of the structure, we reviewed both the software documentation and the source code.

As part of the structural analysis, we identified concurrent program threads and checked for safe exchange of data between threads and absence of deadlocks.

From the architecture analysis, we identified the variables that passed information between the threads. It is important that these updates are “atomic”, i.e. cannot be seen by another thread in a partially updated state. For example, if a two byte variable

is updated from “00, FF” to “01, 00”, it could transiently have the value “01, FF” or “00, 00” (depending on the update order of the bytes). Clearly it is undesirable that the variable can be seen in this transient state, and the software should ensure that all updates are atomic. The software was manually reviewed to check for this.
We also verified that these atomic updates do not result in deadlock. Whenever interrupts are set, they are always released.

6.3 Code Integrity Assessment

Integrity static analysis [10] focused on structural faults in the software, in particular the internal integrity of the code and the intra-component integrity.

The extensive field experience of the device being analysed is likely to precipitate the detection of most large and obvious faults during typical execution of the program. However, specific vulnerabilities of the languages used have a less frequent manifestation and are more likely to remain undetected. Integrity static analysis focuses on what we called intrinsic faults, i.e. faults that may be recognised as such independently of any requirements specification. This includes use of out of bound array indexes, use of illegal pointers, use of non-initialised variables, violations of assertions, permanent loss of resources, insufficient resources, dead locks, non-deterministic behaviour and non-controlled access to shared variables.

Our approach to the analysis of the assembler code can be grouped into two main categories:

- Direct analysis of the assembler code, including analysis of the control flow supported by model checking, dead code checks and semantic analysis.
- Translation of the assembler code into C (by a Perl script combined with manual translation) and analysis of the translated code using CodeSurfer and Safer C.

6.4 Redundant Software Analysis

The software was reviewed for redundant functions. The only redundant function found was a display function that apparently drives a local LCD display on the sensor—however the sensor supplied to us has no LCD display. We suspect that this is because the same code is used to drive a family of sensors including some with built-in displays. From a configuration management and support perspective, it is desirable to use common code. There is no real objection to this apart from the fact that it must be demonstrated that the software does not “hang” waiting for a response from the display hardware and it conflicts with IEC 60880, a fundamental standard for reactor protection systems. We do not think this is a problem as the unit supplied to us functions correctly without the LCD display yet a more formal safety justification would need to document this and to justify the non-compliance with IEC 60880.

6.5 Failure Integrity Analysis

It is important that failures of the hardware and the software result in a safe state. An analysis was performed to identify what fault detection and fault handling features were present in the software. Analysis showed that internal software failures and delays were trapped by a hardware watchdog. An analysis of the software also showed that it contained checks on the integrity of the hardware (as required in IEC 61508 Part 2 Annex A1). These included software checks for breaks in probe wiring, memory integrity, and the integrity of the analogue to digital interface. Failure integrity was also enhanced by using a software-generated oscillating binary output to drive the analogue output circuit. If failures occurred in the processor, output interface, or software, the output bit would become static and result in an out-of-range output signal.

6.6 Functional Analysis-Accuracy Analysis

The functional software analyses focused on checking whether the software maintains sufficient accuracy. The functional analyses were performed using the Cosmic suite of tools for assembling, simulated execution, debugging and profiling of the code for a micro-controller.

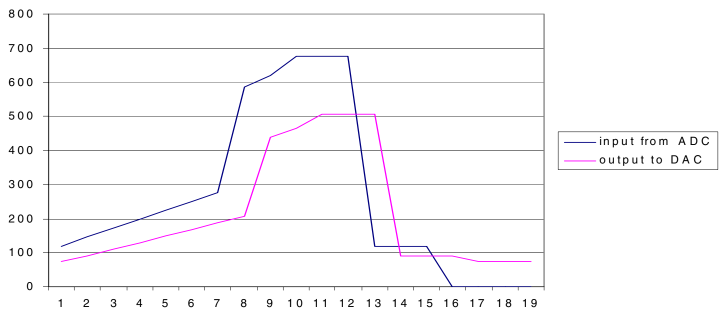

Fig. 2. Input vs. simulated output (mV operation, no linearisation)

The smart sensor software can be configured to operate a number of different modes, or types of measurement. We configured the software to perform a simple linear scaling of the input, and then simulated the execution of the smart sensor code on a PC using the COSMIC simulator tool.

Figure 2 plots the simulated input (X) alongside the converted output (Y). (The units along the horizontal axis count the samples.) The plot confirms a precise match between expected and actual behaviour: the output closely follows the overall envelope of the input, albeit with its slope adjusted by a factor of 0.75, until the input is plunged to zero. The final drop in the input simulates an error condition such as a broken wire or other malfunction of the analogue input. When such errors occur, the output should be forced to the minimum output signal value of 75—and this behaviour is observed in the simulation. Figure 3 plots the magnitude of the rounding error for the first 15 samples in the input sequence. The rounding error is defined as the difference between the value produced by the conversion equation and the "ideal" value.

The plot shows the rounding error to be bounded within +/-0.5 of one output unit. This represents an error in the order of 0.0125% compared to the output of an "ideal" analogue converter. This is in line with our expectations based on inspection of the code analysis, which showed that high precision arithmetic routines are used in the conversion calculation.

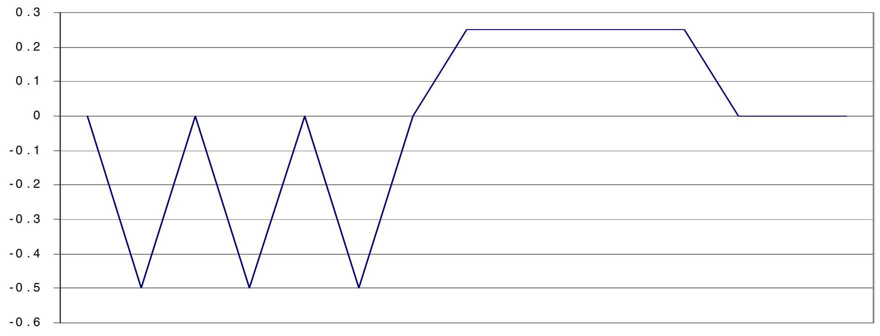

Fig. 3. Magnitude of rounding error

6.7 Timing Analysis

It is important that the sensor has predictable timing, so the measurements are updated at the specified intervals. Inspection of the code showed that the input-output conversion was performed in a simple cyclic loop. This helped to ensure that the sampling of the input and updating of the output occur at predictable rates.

The overall strategy followed by the analysis was to:

- Obtain simulated execution timings (using the COSMIC tool) for the software conversion.
- Derive worst-case estimates for the amount of time the micro-controller spends communicating instructions and data with the ADC. These are derived based on timing data available on the serial interface between the micro-controller and the ADC, as well as the ADC's internal operation.
- Combine the two, bearing in mind potential overlaps of activity, and check the combined timings against the claimed targets.

The complete timing estimates were derived from the information above to substantiate the original claims, and it was concluded that the software performs within its timing specifications, even under conservative and pessimistic assumptions.

7 How Evidence Fits the Safety Justification Approach

This section relates the analysis evidence obtained from the sensor (Section 6) with the goal-based approach (Section 4) and black-box concerns (Section 5). Note that the overall "triangle" of safety justification also included an assessment of compliance to the British Energy PES guidelines [2]. However, standards compliance will not be discussed further in this paper.

7.1 Goal-Based Approach

A goal-based approach for a smart sensor would focus on demonstrating key properties of the sensor that are relevant to the safety justification of the system. In Section 4 we identified a set of sensor properties that represent the top-level goals of the smart sensor justification. This section discusses the evidence that can be used to show that the goals have been met.

- Evidence for goals 1 and 2 (accuracy and response time) can be obtained from black-box functional tests (e.g. SIREP or statistical tests). However, it would also be possible to strengthen the justification by using diverse evidence from white box analyses. For example, the analyses of software accuracy and timing undertaken in Sections 6.6 and 6.7 could be part of an alternative white-box argument that the overall combination of hardware and software will satisfy the top-level goals.
- Goals 3 and 4 (safe failure fraction and MTBF) can also be demonstrated by diverse means. With adequate reporting of operational defects by end users, it might be possible to analyse the field experience to directly measure the MTBF and safe failure fraction. However, it is recognised that there are many uncertainties in such estimates, and it is also possible to make an analysis of the hardware and diagnostic features to make alternative estimates. In our particular example, the supplier provided a FMEDA assessment to support the claims.
- Goal 5 (configuration and calibration integrity) is important as errors in configuration and maintenance are a significant source of safety problems. Clearly the configuration interface should seek to minimise errors. We have analysed the design features that protect against random corruption using evidence from the integrity analysis, but this does not cover aspects such as the usability of the interface. This might be addressed by a human factors assessment.
- Goal 6 (predictable behaviour) is generally important within the safety justification. If the behaviour is predictable, the results of black-box tests can be used with more confidence because the behaviour is likely to be repeatable. To some extent predictability can be justified by field experience, but most of the evidence is likely to be based on analysis of the architecture and source code. The structural analyses described in Section 6.2 give confidence that some known architectural problems (like deadlocks) do not exist, and the vulnerability assessment summaised in Section 7.2 also help to demonstrate predictability. In addition, the integrity analyses we performed showed that certain types of source code faults were absent. These are however quite detailed analyses that may be impractical for lower integrity sensors, so an alternative source of evidence might be grey-box knowledge of the design process. For example, the process might include rules about the design of the system for deadlock avoidance and predictable timing or procedures to perform static analysis to detect various types of software fault.
- Goal 7 (long-term dependability) is also important for the long-term safety of the overall system. Smart sensor products are subject to change (e.g. of hardware components and software functionality) and might also cease to be manufactured or supported. We need evidence of availability and support, but we also require assurance that changes do not affect any of the other safety goals. This is partly addressed by the technical analysis given in Section 6, but knowledge of the supplier's process can give some confidence that the changes will do not affect the behaviour of the sensor.

It can be seen that a goal-based approach helps to identify what evidence helps to support the safety justification. We can also see that multiple forms of evidence could be deployed to support the same goal, and this gives flexibility in the justification approach. With higher integrity applications, we might expect greater use of diverse evidence and arguments. We also note that evidence about the process is only indirectly related to the functional properties of the sensor, but is likely to be highly relevant in ensuring that the observed behaviour is predictable and will be maintained in subsequent upgrades.

7.2 Vulnerability Assessment

This section summarises how specific concerns have been or could be addressed to support the justification of a smart device. These would support goal 6 in Section 7.1.

Table 2. Summary of vulnerability analysis

Concern	Activity/method	Observation
Adequacy of functional testing	Functional analysis	Partially tested through simulation to establish functional requirements and accuracy were met.
Housekeeping code	Code inspection & structural analysis	No problems identified.
Detection of infeasible paths	Dead-code analysis, simulation	Dead-code analysis identified few cases of inaccessible code. Simulation uncovered no infeasible paths.
Fault-detection code	Code inspection and structural analysis	Inspection revealed no problems.
Fault-tolerance	Code inspection and structural analysis	The design of the product emphasises fail-safety over fault-tolerance. The software has good fault integrity behaviour.
Time-based events	Code inspection, timing analysis	No calendar time-based events were found.
Counters	Structural analysis, code analysis	No cases of potential counter overrun were found in the code.
Malicious code	Code inspection	None found.
Unwanted requirements	Documentation review, code analysis	Some unused code was found, but it was used by other models in the same sensor family.
Security vulnerabilities	Code inspection	None found in the software running on the product itself.
Complexity	Structural analysis, code inspection	The software architecture is simple. Main sources of complexity are the relatively dense branching structure of the code and the bit-field encoding of configuration data.
Concurrent interactions	Structural analysis, concurrency analysis	The sequence of functions performed is predictable. Concurrency analysis shows that data exchange between threads is limited and the data exchanges are atomic.
Initialisation errors	Structural analysis	Not analysed in detail.
Data overflow	Structural analysis	Not analysed in detail. However, most of the data used are fixed multi-byte variables, so overflows are unlikely.
Variable time response	Structural analysis Timing analysis	Analyses have shown the code will execute within the specified response time.

8 Conclusions

In this section we present the conclusions of the research study into methods for justifying smart sensors that this paper is describing. While some of the conclusions follow directly from the work here described, others have arisen from work not specifically mentioned in the paper. The conclusions of the project are given below.

8.1 Relationship with Sensor Manufacturers

- End users should expect different degrees of co-operation, both within a single manufacturer's organisational hierarchy and across different manufacturers.
- It remains an open question whether end users will be offered as much access to software as we have in the course of this project. However, the level of co-operation might increase with the potential of a real order, which would not be the case in a research project.
- We need to be specific about what needs to be demonstrated in an assurance package that is provided by a supplier. Some form of assurance about the product is needed, principally to confirm predictability of behaviour and integrity in the presence of hardware failures. This evidence would typically be a combination of "grey box" (e.g. documentation on the software design) and white box evidence (based on analysis of the actual software), as greater confidence could be obtained if there was some knowledge of the internal design and implementation of the device.

8.2 Safety Justification Approaches

- Justification may follow a goal-based approach, aim to demonstrate compliance of the product with industry guidelines, or seek specifically to address areas of concern that a typical black box assessment would not cover. These three viewpoints should be considered simultaneously as they complement each other.
- Goal-based approaches seem particularly suitable to smart sensor products as they focus directly on the safety requirements, can be tailored to standards and offer flexibility and a certain potential for reuse of arguments and evidence.
- We have addressed specific concerns about the shortcomings of black-box assessment by assessing their applicability to smart sensor products similar to the ones we have examined. Grey-box and white-box evidence can compensate for the limitations of black-box assessment.

8.3 Smart Sensor Evaluation Methods

- We have undertaken a range of structural, accuracy and timing analysesbased on documentation and the source code, and these have proved to be "easy wins".
- Such analyses might be improved with appropriate tool support (e.g. code and data flow dependency analysis, worst case time analysis), although this might be problematic for assembler based systems.
- We have shown that it is feasible to simulate execution to test specific attributes such as accuracy, test coverage and timing. However it is likely that these tests would have been easier to implement during development where there is no need to simulate the action of attached peripherals.
- It was technically feasible to perform a range of evaluations on the smart sensor (even though it was written in assembler). This was largely due to the relatively small size of the software and simplicity of the design.

8.4 Further Work

There are many pragmatic issues that have not been addressed within the current study (e.g. what evidence is needed for a given integrity level and who should perform the evaluation). However we hope that the feasibility studies performed in this project can contribute to the development of a common approach to the justification of smart sensors in the nuclear industry.

Acknowledgements

This work was partly funded under the HSE Generic Nuclear Safety Research Programme under contract 40072849. The views expressed in this report are those of the authors and do not necessarily represent the views of the members of the CINIF or the Health and Safety Commission/Executive. The CINIF does not accept liability for any damage or loss incurred as a result of the information contained in this paper.

References

[1] Nuclear Safety Directorate, "Safety assessment principles for nuclear plants", http://www.hse.gov.uk/nsd/saps.htm
[2] L.A. Winsborrow, A.R. Lawrence, "Guidelines for Using Programmable Electronic Systems in Nuclear Safety and Nuclear Safety-Related Applications", British Energy 2002.
[3] P.G. Bishop and R.E. Bloomfield, "The SHIP Safety Case—A Combination of System and Software Methods", SRSS95, Proc. 14th IFAC Conf. on Safety and Reliability of Software-based Systems, Brugge, Belgium, 12-15 September 1995.
[4] P.G. Bishop and R.E Bloomfield, "A Methodology for Safety Case Development", Safety-critical Systems Symposium, Birmingham, UK, Feb 1998.
[5] CEMSIS project, http://www.cemsis.org
[6] J.A. McDermid, "Support for safety cases and safety argument using SAM", Reliability Engineering and Safety Systems, Vol. 43, No. 2, 111-127, 1994
[7] C.C.M. Jones, R.E. Bloomfield, P.K.D. Froome and P.G. Bishop, "Methods for assessing the safety integrity of safety-related software of uncertain pedigree (SOUP)", Report No: CRR337 HSE Books 2001 ISBN 0 7176 2011 5 http://www.hse.gov.uk/research/crr_pdf/2001/crr01337.pdf
[8] P.G. Bishop, R.E. Bloomfield and P.K.D. Froome, "Justifying the use of software of uncertain pedigree (SOUP) in safety-related applications", Report No: CRR336 HSE Books 2001 ISBN 0 7176 2010 7 http://www.hse.gov.uk/research/crr_pdf/2001/crr01336.pdf
[9] P.G. Bishop, R.E. Bloomfield, T.P. Clement and A.S.L. Guerra, "Software Criticality Analysis of COTS/SOUP" Safecomp 2002, Catania, September 2002.
[10] P.G. Bishop, R.E. Bloomfield, T.P. Clement, A.S.L. Guerra, and C.C.M. Jones, "Integrity Static Analysis of COTS/SOUP" Safecomp 2003, Edinburgh, September 2003.