Data Management in Data-Driven Safety-Related Systems

Neil Storey, B.Sc., Ph.D., FBCS, MIEE, CEng;
School of Engineering, University of Warwick; Coventry, UK

Alastair Faulkner, M.Sc., MBCS, CEng;
CSE International Ltd.; Flixborough, UK

Keywords: data, safety-related systems, software, safety

Abstract

The increasing use of COTS components is leading to the production of a large number of systems which use standardized hardware and software that are customised for a particular situation by the use of configuration data. Where such systems are used in safety-related applications, the safety of the resulting system will often be dependent on the correctness of this data. It is therefore essential that configuration data is developed and tested to the same level of rigour as other system elements.

Despite the obvious importance of data correctness in safety-related systems, anecdotal evidence suggests that data does not receive the same attention as other system elements. This view is reinforced by the observation that the standards in this area say almost nothing about the design, production, verification or maintenance of data.

This paper describes a study to investigate the techniques being used to produce and manage data in a range of safety-related industries. This shows that data is indeed being largely ignored in many highly critical situations. The paper then goes on to suggest a way of tackling this problem, through the development of generic, and perhaps industry-specific, guidelines.

Introduction

Over the last decade we have seen an increasing reliance on computers in all forms of safety-related system. Because of the high development costs associated with such systems, developers have moved towards the use of standardised hardware and software wherever possible. This in turn has led to the large-scale use of COTS products, and to the development of systems that can be easily adapted to a range of similar situations [ref. 1].

COTS and multipurpose systems are often adapted to a particular installation through the use of configuration data. In some cases, this configuration data is extensive, and represents a substantial part of the complexity and the cost of the complete system.

Since safety is a property of a complete system, rather than its individual components, it follows that attention must be paid to *all* its components during the development process. Over the years a great many techniques have been developed for treating the hardware and software elements of computer systems, but less attention has been directed at the data element. This is evident from the various standards and guidelines relating to safety-related systems. Generic standards, such as IEC 61508 (ref. 2) provide extensive guidance on hardware and software issues, and are used across a range of industrial sectors. However, these say almost nothing about the generation, testing or control of data. Similarly, industry specific standards in the civil aircraft, military, nuclear and railway sectors (refs. 3 – 6) give very little guidance in this area.

Perhaps one of the reasons why data is given so little prominence within these documents is that they make the tacit assumption that data is simply part of software and is therefore covered by the general guidance given. Certainly, many definitions of software include data (and documentation) within its remit. However, it is quite clear that in many cases configuration data is not developed alongside more conventional software, and is not developed with the same degree of care.

To see how and why data is treated differently from software we need to look at the nature of data within computer-based systems. Here we are primarily interested in systems that make extensive use of large amounts of data. We will refer to such systems as data-driven systems.

Data in data-driven systems

All computer programs make use of data. Most programs will contain constants that are used within their operation, and will calculate values as they are executed. Temporary, or intermediate values, may be stored in specific memory locations or on a stack. Many systems also make use of calibration constants or device characteristics for particular elements. All these can be thought of as either constant or variable *data*.

While all programs make use of some form of data, some make much more extensive use of data than others. In particular, some rely on large amounts of data to configure the system for a particular situation. A characteristic of such systems is that the configuration data is normally generated quite independently from the 'executable' part of the software.

In many data-driven systems, some form of application software implements a series of required functions that are then applied to a set of data. The application software may be a COTS program, or some custom or semi-custom software. The data on which it acts often represents some aspect of the physical world. For example, the application data could be a railway control system and in this case the data would represent (amongst other things) the physical layout of the tracks and the instantaneous positions of the trains. In this example the data is used to configure the system for this particular situation. The same applications software could be used with different data to control a different railway network in another location. Other examples of data-driven systems include those used in air traffic control. Here commercial ATC packages are configured to work within a particular air space by the use of data. This data will include a fixed description of the area (including an altitude map of the region and the location of airfields, etc.) and transient data on the positions of aircraft in the region at any time.

In these examples, and in many data-driven systems, the application software is often developed and supplied together with the hardware of the system. This software may well be a standard product that has been developed and refined over a number of years. However, the configuration data is invariably a unique set of data that is developed quite independently.

Data development

While the configuration data is often developed separately from the application software, it is clearly an integral part of the complete system. Consequently, the correctness of this data will normally be closely linked to overall system safety. This being the case, it is likely that the safety integrity requirements of the configuration software will be comparable to those of the application software.

In any safety-related application the development methods used must reflect the safety integrity level (SIL) of the system. If the SIL of the configuration data is the same as that of the application software, then one would expect that each would be developed and verified with a similar amount of care and rigour. However, there is much anecdotal evidence to suggest that this is not the case.

A possible reason why data might be treated differently from software, is that very little guidance is available on appropriate data development methods. Within most standards and guidelines data is considered to be an integral part of software. However, while these documents give a great deal of attention to the executable aspects of software, they give little attention to the special requirements of data.

As an example, if we look at IEC 61508 we find that a complete section of the standard relates to software. The section contains a subclause entitled 'General Requirements', which sets out a range of requirements for the software of the system. This section ends with the statement "This subclause … shall, in so far as it is appropriate, apply to data including any data generation languages." This statement perhaps sums up the treatment of data within the standard. Data is seen as an integral part of the software, having few special characteristics or special requirements.

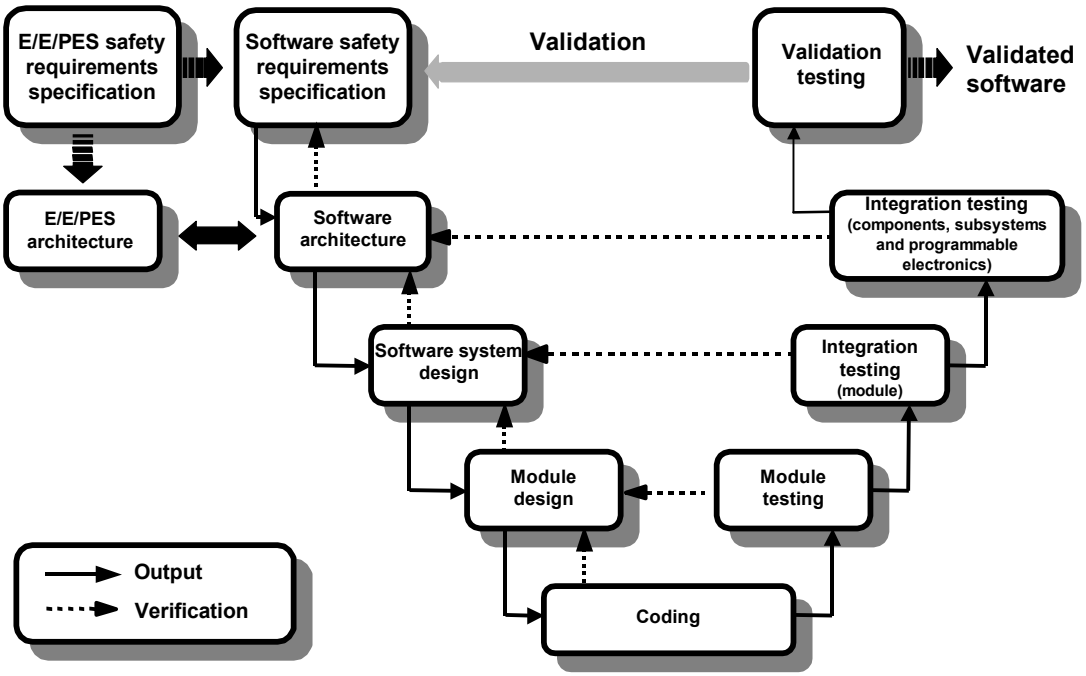

Figure 1 – Software safety integrity and the development lifecycle – from IEC 61508

It would be unfair to suggest that IEC 61508 says nothing about the requirements of data. For example, it says at one point that "…the design method chosen shall possess features that facilitate … the expression of … data structures and their properties". However, the standard says nothing of the methods that should be used to generate or verify data, and it does not identify specific data requirements.

A data development lifecycle

The authors of this paper would argue that since data in data-driven systems is often developed separately from other parts of the system, it should be treated separately and have its own requirements documents and its own development lifecycle. A reasonable starting point in the development of such a lifecycle model might be the software development lifecycle given within IEC 61508 and shown in Figure 1. This is largely the standard 'V-model' lifecycle, with which most engineers will be familiar. For those not acquainted with this standard, the acronym E/E/PES stands for electrical/electronic/programmable electronic system. One could argue that a separate lifecycle for data is unnecessary since it is already covered by the existing software lifecycle. A powerful argument for this point of view would be if it could be shown that data was currently being developed in line with the lifecycle of Figure 1. This would suggest that further guidance or regulation was unnecessary. However, if it could be seen that data was currently being ignored, it would seem to strengthen the authors' claim that a separate lifecycle model would have benefits.

A survey of data development methods

In order to investigate current development methods for data, a series of structured interviews were carried out, with representatives from a range of industries.

The interviews were confidential, to allow for frank descriptions of the company's operation. Although the companies cannot be identified here, the industries/sectors involved were as follows:

- Railway command and control systems
- High integrity railway interlocking
- Underground railway systems
- Process control systems
- Air traffic control systems
- Urban road traffic signalling systems
- Electrical engineering systems
- Safety regulator

In all cases the people questioned were senior engineers with many years of experience within their industry. They were therefore able to give an authoritative view on the development methods being used within their organisations

The structured interview.

Table 1 shows the questions asked within the structured interviews. Each interview took a couple of hours, and sufficient time was allowed to gain a reasonable appreciation of the issues covered, although clearly not all questions were relevant in all cases.

1	Does your organisation use, or have an interest in, data-driven safety-related systems?
2	Please identify these data-driven safety-related systems and outline the safety management process used to bring them into service.
3	How are data-driven safety-related systems selected for use within your industry?
4	How will the data-driven systems(s) be affected by failures of data, and what types of data failure have you considered?
5	How are data structure and data content issues identified during the design process?
6	What factors were considered in deciding whether or not to adopt a COTS solution? Do you consider that your decision took into account the full cost of ownership of COTS?
7	What motivated you to adopt a data-driven system?
8	How has the use of a data-driven system affected the safety approval or certification process that you apply?
9	What do you consider to be the cost implications of the use of data-driven systems?
10	What data does your data-driven system use?
11	How is the configuration data validated?
12	Does the configuration data lend itself to off-line validation by automated tools?
13	Does the generic application software perform any internal (on-line) checks on the validity of the configuration data?
14	How will configuration data be collected and by whom?

Table 1: Questions within the structured interview

Many of those interviewed said that the interview had raised issues that they had not previously considered, and found the process helpful in stimulating thought in this area.

Results from the interviews

The survey produced a great deal of information on the development of data-driven systems, and demonstrated, as expected, that these are often developed differently from more conventional safety-related systems. It would not be appropriate to discuss all the findings of the survey within this paper, although these will be published elsewhere. Here we will concentrate on the key points raised by the survey.

One of the strongest, and perhaps most significant points, is that hazard analysis is often not applied to data in the same way that it is applied to other system elements. This means that the possible failure modes are not identified and the consequences of such failures are not studied.

Similarly, it is common not to apply integrity requirements to data. This omission masks the need to demonstrate that data has been developed to an appropriate level of quality.

The survey also suggests that the development process changes significantly with time. When a data-driven system is applied for the first or second time, the developers tend to be involved in this process and to contribute directly to the work involved. The developers are therefore able to give guidance on the implementation methods used, and on the 'intent' behind its construction. In a sense these early applications form part of the final validation of the system, confirming that it satisfies its requirements in an actual application.

Providing that these early implementations are successful the developer's involvement in the installation process will normally cease. Later applications will not have the benefit of the input from those who created the system and will need to rely on documentation and their understanding of the developer's intent. As the system is more widely applied, its ability to be tailored to a given situation may result in it being used in circumstances that were not envisaged by the original design team.

It could be argued that each new applications of a general-purpose data-driven system is a unique system and that each should be independently verified and validated in full, to ensure that it is safe and that it satisfies its requirements. However, in practice this is not done. Justification of the system is normally based on the assumption that the configuration data represents an essentially independent module within the system. Since the remainder of the system has been validated in another situation, it is assumed that changing the data only requires that the data be separately validated.

Unfortunately, while it would be possible to design a system to allow the data to be seen as a separate module that could be validated independently, the survey shows that this is often not the case. Many applications rely on data that appears to have no modular structure and that is used in a very poorly defined manner. Under these circumstances small changes to the data can have significant, and unpredictable effects on the system. In such cases even minor changes to the data would seem to require that the entire system be retested and validated – not just the modified data.

Another problem is that the data associated with many systems is not in a form that allows it to be validated separately from the complete system. The survey shows that data often does not have defined requirements, structure or function that would allow it to be validated as a separate entity. In such situations it can only be investigated as part of the complete system. In these circumstances one can only determine the 'safety' of a unique set of data by investigating (validating) the complete system.

An interesting finding of the survey is that data-driven systems would seem to be more likely to be used in situations where they interact with other data systems. They therefore tend to contain data interfaces to other systems within an organisation. Data passing through these interfaces poses its own unique problems for system integrity and safety. One only has to think of examples such as railway signalling or air traffic control to see how safety is reliant on the quality and timeliness of such data.

A final observation is that data-driven systems tend to be larger in scale that other bespoke systems, increasing the need for effective configuration management.

Conclusions from the survey

The findings of the survey suggest that data within data-driven systems is often *not* being developed in the same way as application software. It also suggests that in many cases the data is not being developed and managed in a way that would satisfy the requirements outlined in IEC 61508, or other standards, for the development of software.

Perhaps the most worrying finding is that hazard analysis is often not applied to configuration data in the same way that it is applied to other system elements. This inevitably leads to a situation where data faults are not identified as significant system hazards.

A failure to identify data-related hazards means that these are not covered by the safety requirements of the system. This in turn means that the system architecture, and system design, will not attempt to deal with these hazards. If one does not identify data faults as a significant hazard, then no effort will be put into avoiding, removing, detecting or tolerating such faults.

Also of concern is the fact that in many cases no integrity requirements are being applied to data. This seems to imply that data faults are seen as unimportant. When an integrity classification is applied to a system element (such as hardware or software) this brings with it an expectation that faults within those elements will be dealt with in a manner appropriate to that integrity level. For example, IEC 61508 has target failure rates for each of the safety integrity levels. A failure to assign integrity requirements to data ignores the obvious importance of this system element.

Another major finding of the survey relates to the verification and validation of systems that have been configured for a particular situation through the use of data. In such situations it is not normal to fully validate the configured system, but to rely heavily on confidence gained by validating the system without the data, or with a different data set. This approach would seem to assume that either the data is a completely separate module that can be investigated separately, or that the safety of the system is not affected by the data. Both of these assumptions would seem to be invalid.

In many cases companies take great care in the production of configuration data, but the techniques used tend to be selected in an unsystematic manner. This may be because little guidance is available within the literature to aid this choice.

Recommendations

It is clear that the management of data in data-driven safety-related systems is not always receiving the attention that it should. For this reason, the authors suggest that more specific guidance is required for this system element. We propose that data should be specifically identified as a system element within standards such as IEC 61508, and that explicit and unambiguous guidance should be given on appropriate methods for its specification, design, implementation, verification and validation.

One way of encouraging engineers to look specifically at the management of data, is to assign to it a dedicated development lifecycle model. Figure 2 shows a possible lifecycle, this being based on the software development lifecycle of IEC 61508.

The lifecycle of Figure 2 does not show hazard and risk analysis as a separate phase since these tasks are carried out throughout the development process. However, the identification of a ‘data safety requirements specification’ will require that hazard and risk analysis be carried out at an early stage in the development process. The data safety requirements will provide an input into the planning of the overall data architecture and may require such features as data fault tolerance.

A key recommendation is that the integrity requirements of the data should be specified explicitly. This will place a requirement on the developer to demonstrate that the data has been developed in a manner consistent with the specified data integrity level. It will also require that the developer shows that data faults have been considered and dealt with appropriately.

More detailed aspects of the design will include consideration of the data structures to be used. Here the form of the lifecycle model encourages engineers to consider the need for verification of the populated structures at a later time. Experience shows that many existing systems use poorly structured data that makes verification impossible. Such techniques would not be tolerated in the production of executable software, and should not be acceptable within data management.

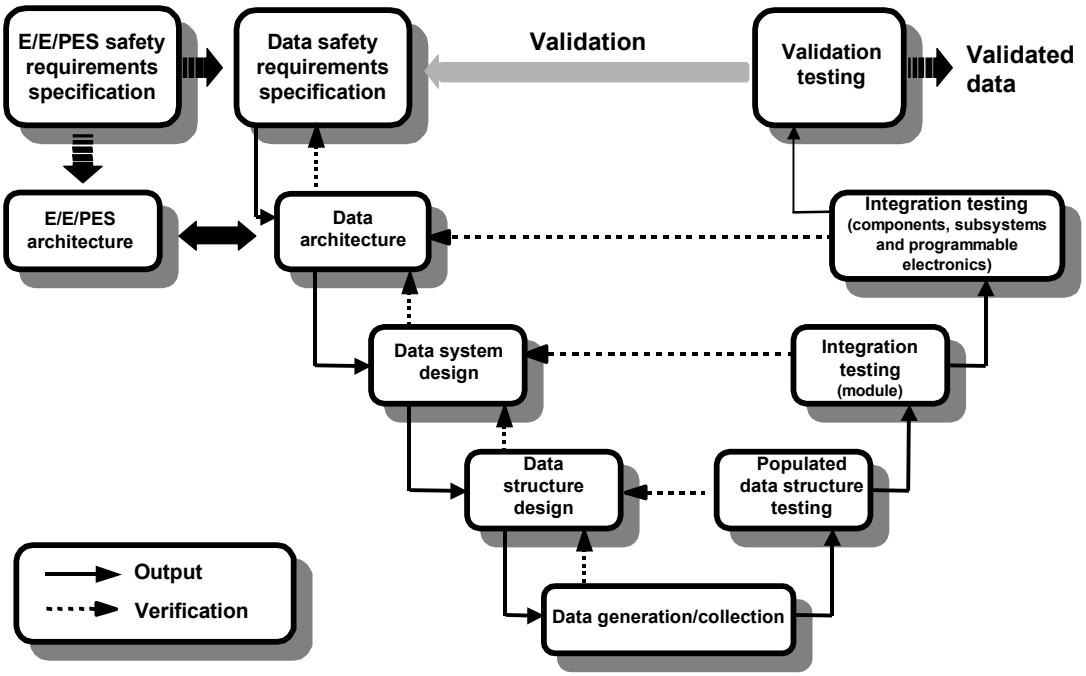

Figure 2 – A possible development lifecycle for a generic application of a data-driven system

The generation or collection of data is clearly an important part of data management (although it is *not* the only part). Care must be taken to ensure that the methods used are appropriate for the safety integrity level of the overall system. In some applications data will be generated or computed automatically, while in others it will be produced manually. In either case thought must be given to the required accuracy of the data and the allowable error rates. It is also vital that the origins of the data are recorded to establish an audit trail for use in subsequent safety justification.

The need for verification of the data is likely to place considerable constraints on the way that it is produced and stored. If one looks at the corresponding situation in the production of executable software, we know that the use of low-level languages is discouraged since they produce poorly structured software that is likely to contain errors and which is hard to verify. In the same way, requirements for verification will require that data is produced and structured in a manner that permits verification.

Software verification makes use of a range of dynamic and static test tools and it is likely that a similar range of tools will be needed for the verification of data. In particular the production of 'static data analysis' tools is seen as being very important. It is likely that such tools will require that data is structured, and perhaps annotated, to allow effective verification.

The production of a data safety requirements specification will automatically influence the validation process, since the developers will need to demonstrate that issues contained within that document are addressed within the implementation.

While the use of a dedicated data lifecycle may assist in the development of a generic data-driven system, it does not tackle the specific problems associated with the adaptation of that generic system to suit a particular situation. One of the issues raised in the survey is that although appropriate verification and validation methods may be used for the generic case, they are not always applied to each application-specific instance of the system. One way of tackling this problem is to have a development lifecycle model for the *application* that is separate from that of the *generic system*. Separating the application from the generic system in this way allows individual guidance or requirements to be applied to the development of these two distinct aspects of the system.

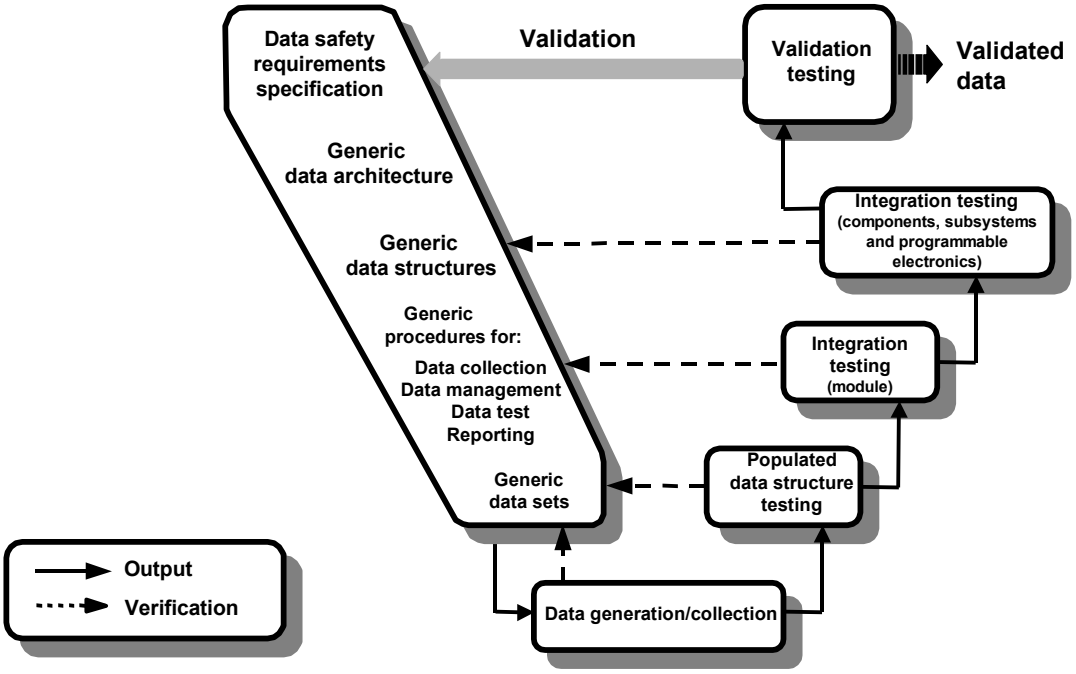

Figure 3 – A possible development lifecycle for an instance of the application of a data-driven system

Figure 3 shows a possible lifecycle model for an application-specific instance of a data-driven system. This removes those activities related to the development of the generic application and emphasises those tasks related to ensuring the safety of a particular application.

The left-hand side of the lifecycle of Figure 3 clearly represents the design elements of the development of the data. In a system formed by configuring a generic application, this work will have been performed at some time in the past. However, it is essential that those responsible for configuring the system have access to the various 'deliverables' from this work. These include the data safety requirements specification, the data architecture and the data structures, but also defined procedures for the collection, management and testing of the data. Other deliverables should include details of an appropriate reporting structure. For example, a data test report should have been created and be awaiting data test results. Some mechanism must also exist to ensure that information gained during the development and use of an individual system is fed back to provide inputs to future development of the generic aspects of the system.

While it would seem clear that the deliverables listed above would be required for the correct configuration of a data-driven system, experience shows that these are not available in many (perhaps most) instances. The use of a well-documented lifecycle model of this type might help to alleviate this problem.

The lifecycle models described above are based on the generic standard IEC 61508. While this is a widely used standard it does not meet with universal acceptance. However, the principles outlined above could also be applied to the lifecycle models used within other standards to produce equivalent ways of describing the process of data management.

It is inevitable that individual industrial sectors will have different requirements and practices for the use and production of data. Therefore while generic standards can give general guidance on matters related to data, industry-related standards will inevitably be needed to give more specific information.

Discussion

The survey in this paper does not represent an exhaustive study of the treatment of data in data-driven systems. However, it does suggest that this is an area that is worthy of further study.

The informal survey of engineers working in the area came up with some interesting points:

- Data-driven systems are often complex in comparison with conventional computer-based systems.
- Data-driven systems often form part of a hierarchy of computers that exchange real-time data.
- This complexity, and the interchange of data, makes data-driven systems challenging to design. It may also make it difficult for those charged with configuring the systems to gain a full insight into the original design intent.
- Current approaches to configuration and validation of specific applications depend on assumptions of modularity and independence of the data.
- Modularity and independence require well-structured data that has well-defined interfaces to the application software. These are often missing from current data-driven systems.
- These deficiencies can produce problems for system validation.

The task of validating a particular instance of a generic system might be simplified by producing a generic safety case that covers the product, and a separate supporting document that relates to a particular application or instance. The latter could then be updated to represent the particular details of an individual installation. The generic part of the safety case would need to argue (and demonstrate) that changes to the data would not affect overall system safety. This might require that such changes could be shown to have a limited scope for affecting the operation of the system and that faults within the configuration data could be adequately detected by testing. It is likely that such an argument would require that the data was adequately structured and validated to ensure its integrity.

Since data-driven systems are often complex, they may benefit from visualisation techniques that offer a high level of abstraction. One approach to this problem uses *patterns* to represent commonly used (and re-used) components. This technique has been applied to software architectures and configuration management, where it can help identify bad architectures or practices (which are known as *antipatterns*). Patterns have also been applied to the reuse of common structures within safety case arguments (ref. 7). Such reuse would seem to be of considerably benefit in applications where a generic system is applied in a range of situations. It is also possible that patterns may have relevance in the development of data architectures and data structures.

Despite the varied approaches used, it is clear that configuration data is treated very differently from application software. This would tend to suggest that it should receive separate treatment within standards and guidelines. It appears that data is often not subjected to the same degree of hazard and risk analysis as other system elements and this may result in data-related hazards being overlooked. It also appears that data is not normally assigned any specific integrity requirement. The use of a separate lifecycle for configuration data, with a specific data safety requirements specification, might overcome many of these problems.

Conclusion

The widespread use of COTS is leading to an increasing number of systems that make use of large amounts of configuration data. A survey covering a range of industries suggests that this configuration data is not always being developed in a manner that is consistent with the integrity requirements of the system in question. One reason for this could be that data is largely ignored within the various generic and industry-specific standards.

This paper suggests that within data-driven systems, data should be considered as a distinct system element with its own requirements documents and lifecycle. Generic standards such as IEC 61508 should also give specific guidance on the design, production and verification of data. This, it is hoped, will encourage engineers to give the production and management of data, the attention it deserves.

References

1. J. A. McDermid "*The cost of COTS*", IEE Colloquium - COTS and Safety critical systems London, January 1998.

2. IEC 61508 Functional Safety of electrical / electronic / programmable electronic safety-related systems Geneva: International Electrotechnical Commission, 1998.

3. RTCA DO 178B / EUROCAE ED-12B Software Considerations in Airborne Systems and Equipment Certification Washington: Radio Technical Commission for Aeronautics, Paris: European Organisation for Civil Aviation Electronics, 1992.

4. Interim Defence Standard 00-55 The Procurement of Safety Critical Software in Defence Equipment. Glasgow: Directorate of Standardisation, 1991.

5. International Standard 880 Software for Computers in the Safety Systems of Nuclear Power Stations. Geneva: International Electrotechnical Commission, 1986.

6. RIA Safety Related Software for Railway Signalling (Consultative Document) London: Railway Industry Association, 1991.

7. Kelly T, McDermid J. *Safety Case Construction and Reuse Using Patterns*. Proc. 16th International Conference on Computer Safety and Reliability (SAFECOMP'97). York. 1997.

Biographies

N. Storey, B.Sc., Ph.D., FBCS, MIEE, C.Eng. School of Engineering, University of Warwick, Coventry, CV4 7AL, UK. Tel. - +44 24 7652 3247, fax - +44 24 7641 8922, e-mail - N.Storey@warwick.ac.uk.

Neil Storey is a Director within the School of Engineering of the University of Warwick. His primary research interests are in the area of safety-critical computer systems. He is a member of the BCS Taskforce on Safety-Critical Systems and has a large number of publications including both journal and conference papers. Neil is also the author of several textbooks on electronics and safety, including “Safety Critical Computer Systems” published by Addison-Wesley.

Alastair Faulkner, MSc., MBCS, C.Eng; CSE International Ltd., Glanford House, Bellwin Drive, Flixborough DN15 8SN, UK. Tel. +44 1724 862169, fax +44 1724 846256 email - agf@cse-euro.com

Alastair Faulkner holds an MSc degree in Computer Science from Salford University and is a Chartered Engineer. His background is in software development mainly concerned with computer based command and control systems. He now works on a large UK Rail infrastructure project. Alastair’s research interests are in the data management of data-driven safety-related systems. He is also a Research Engineer with the University of Warwick and is studying for an Engineering Doctorate.