Data Requirements for Data-Intensive Safety-Related Systems

Neil Storey, Ph.D.; School of Engineering, University of Warwick; Coventry, UK

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

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

Abstract

Many computer-based systems make use of large amounts of data to determine their operation. This may be to allow dedicated control systems to be adapted to a particular situation, or to allow standardised hardware and software elements to be configured for a specific application. This latter category includes many COTS products that are configured for a given purpose through the use of data.

While data is clearly crucial to the correct operation of these data-intensive systems, there is much evidence to suggest that it is often overlooked. Data is often not subjected to systematic hazard analysis and consequently data-related hazards are often not identified. For this reason, data in such systems is not normally assigned any specific safety requirements and is not given a particular integrity level.

This paper looks at the characteristics of data in data-intensive safety-related systems and describes the need for dedicated data safety requirements. It then considers both general and specific data requirements in more detail, and discusses issues of data quality and data integrity. It then outlines the data supply process before considering the verification and validation of data-intensive systems.

Introduction

All computer-based systems make use of data in one form or another. This might be in the form of simple constants within programs or temporary values used during calculations. However, some systems use of large quantities of data that determine the fundamental operation of the system. Examples include air traffic control systems, railway signalling systems and battlefield management systems. We are also seeing increased use of systems that use standardized hardware and software that are configured for a particular application through the use of data. These may be conventional COTS applications or custom systems that are to be used in a wide range of situations. All these applications share the basic property that the operation of the system is strongly influenced by, if not determined by, data. These systems commonly operate within a hierarchy of systems that share data, and may be referred to as '*data-intensive*' systems.

When data-intensive systems are used within safety-related applications the safety of the overall systems is determined by the properties of all its constituent parts. Clearly the dependability of the hardware and software elements will be crucial to the safety of the system, but the correctness of the *data* is also essential if the system is going to achieve overall safety. Unfortunately, experience shows that while considerable effort is applied to making sure that the hardware and software components achieve an acceptable level of safety, the data within such systems is often overlooked [ref. 1]. One reason for this oversight may be that the various standards in this area do not identify data as a distinct system element but rather consider it to be an integral part of software. While this is appropriate for the constants and variables found within conventional programs, it is not helpful when dealing with data-intensive applications. In such systems the data often forms a distinct and separate entity that is invariably developed and maintained separately from the software of the system. Data also has very different properties from executable software and requires different techniques for its production and verification. At present the various standards give almost no guidance on the generation, validation or maintenance of data [refs. 2-6].

If data is indeed a major element within data-intensive systems, one would expect that the development methods used to produce this data would reflect the same degree of care and attention that is applied to the other components of safety-related systems. However, experience suggests that this is not the case. To investigate this situation a survey was performed to investigate the techniques being used to develop data within a range of industrial sectors [ref. 7]. The key findings of the survey are that:

- Data is often not subjected to any systematic hazard or risk analysis.
- Data is often not given any specific safety requirements.
- Data is often not assigned any specific integrity requirements.
- Data is often poorly structured, making errors more likely and harder to detect.
- Data is often not subjected to any form of verification.

Many of the problems identified within the survey follow from the first point listed above. Since data is not identified as a specific entity it is often ignored within the process of hazard and risk analysis. This may be because it is ‘hidden’ within the software element of the system, or because data creation is seen as part of system commissioning rather than system design.

If data is not considered within the hazard analysis stage, no data-related hazards will be identified. This will suggest that the data has no specific safety requirements and that little effort needs to be expended in getting it right. This is perhaps the reason why no specific integrity requirements are normally assigned to the data.

As data commonly does not have any specific integrity requirements, little effort is applied to its design or production. Consequently, data is often poorly structured, making data errors more likely and harder to detect. Hardware or software elements requiring high integrity will often make use of fault detection or fault tolerant techniques to overcome faults. Since data invariably has no integrity requirements, such techniques would not normally be considered.

Since data often has no specific safety requirements and no integrity requirements, it is common not to verify the correctness of the data. When the completed system is validated this will clearly provide some validation of the data used in this implementation of the system, but will give little confidence in the safety of other installations that use different data sets.

Data within safety-related systems

This paper is *not* concerned with data that forms an integral part of the algorithms within software components, nor with data faults that may be produced as a result of failures within these algorithms. These topics are relatively well understood and are covered within the literature and the various standards in this area.

In data-intensive applications, data is used not only within the various software algorithms, but also to define and modify the operation of the system. Data may be used in a number of ways in such systems, and the role and characteristics of the data vary considerably. In some applications data will be used to provide a static description of the environment in which the system operates, while in others it will describe the system's configuration. Changes in the environment will give rise to dynamic data that will enter the system through interfaces to other equipment or systems. Data may also be used to store temporal information by holding historical data (which may be used for trend analysis or maintenance purposes) or information on future events (as in the case of schedules or timetables).

In many cases one of the major motivations for adopting a data-driven approach, is that it allows the system to be easily adapted in response to changes in the configuration of the system or its environment. Data-driven systems offer the opportunity to modify the data component and hence influence the behaviour of the system without changing the hardware and software components. However, this ease of modification brings with it potential pitfalls. In *non-critical* systems one of the motivations for using a software-based approach is the ease with which software can be modified to produce system upgrades. In *critical* applications this apparent advantage is illusory, since the effort involved in re-validating the system following modifications to the software, is often much greater than would be required for simple hardware changes. At present many systems use a data-driven approach specifically because this permits simple upgrades through changes to the data. However, this choice may well have been based on the assumption that these changes may be made without any need for re-validation of the data or the system as a whole. Unfortunately, unless the system has been specifically designed with this objective in mind, this assumption is almost certainly unjustified.

Since data plays a large part in determining the behaviour of a data-driven system, any changes to the data must be investigated to see if they influence the safety of the system. This might suggest that any changes to the data would require that the entire system is re-validated. However, modularisation can greatly reduce the amount of re-validation that is required. From our experience of the use of software within safety-related systems we know that the use of an appropriate software architecture can greatly simplify software maintenance. Careful partitioning of software into well-designed modules can allow modifications to be made to one module with only limited re-validation of other parts of the system. A similar approach can also be applied to data, where use of an appropriate *data* architecture can be used to produce a system where minor data modification need not require extensive re-validation.

The data architecture of a system is described by the *system data model*, which should include all the data used by the system. This will include *static* data used to configure the system and *dynamic* data passed to the system through interfaces to other systems. Data maintenance is strongly influenced by the structure and modularity of the data architecture. Wherever possible, data that is subject to change should be placed within separate modules that have limited and well-defined interfaces to the remainder of the system. If this is done, it may be possible to establish safety requirements for the data within these modules, and to demonstrate that *any* data satisfying these requirements will not compromise the safety of the system. This would permit data within such a module to be validated (by demonstrating that it meets its safety requirements) without having to re-validate the entire system. Unfortunately, in many cases it may not be possible to produce a set of data requirements for a single module that will guarantee system safety. In such cases the modification of data within one module will require more extensive investigation of other parts of the system. Where a system is based upon a monolithic data architecture, any data change may require that the entire system be re-validated.

The safety requirements associated with data relate not only to its functional characteristics but also its integrity. It is therefore appropriate to look in more detail at the nature of data requirements.

Data requirements

Data requirements, like other system requirements, have many components. Clearly the data will need to satisfy the *functional data requirements* that relate to the function that the overall system is designed to perform. Additionally, there may be various *non-functional data requirements* which are not directly related to the function of the system, but which bestow beneficial characteristics. For example a non-functional data requirement might be that the size of the data set is less than some particular value, so that it will fit within a certain memory device. A third group of requirements relate to the safety of the system and impose limitations on the data in order to avoid unsafe operation. These *data safety requirements* may then be divided into what might be termed 'specific' and 'general' aspects. The *specific data safety requirements* are related to the function of the system and reflect the effect of particular data items (or groups of items) on the safety of the system. For example, there might be a requirement that a data value representing the separation between two aircraft should always be positive, as a negative or zero value corresponds to an accident. These requirements are very application specific and are generated as a result of data-related hazards identified during the hazard analysis phase. The *general data safety requirements* represent more generic requirements relating to individual data items, groups of items or the complete data set. These include considerations of *data quality* and *data integrity*. While these general requirements will vary considerably from one application to another, the principles involved have many common elements. For this reason the remainder of this paper will concentrate on these *general data safety requirements*, and in particular on consideration of *data quality requirements* and *data integrity requirements*.

The distinction between 'data quality' and 'data integrity' is subtle and workers in this area are not consistent in the way they use these terms. Within this paper we use the term *data quality requirements* to describe those aspects of data that relate to its ability to meet the requirements of the system. These include considerations such as accuracy, resolution and timeliness. In contrast, *data integrity requirements* are taken to represent the importance of the correctness of the data. This will be determined by the likelihood that data errors will cause a system failure, and by the consequences of that failure.

Data quality requirements

One of the few standards that provide guidance on data is RTCA DO 200A [ref. 8], which is concerned with aeronautical data. This identifies a number of aspects of data quality:

- The *accuracy* of the data.
- The *resolution* of the data.
- The confidence that the data is not corrupted while stored or in transit (*assurance level*).
- The ability to determine the origin of the data (*traceability*).
- The confidence that the data is applicable to the period of (its) intended use (*timeliness*).
- All of the data needed to support the function is provided (*completeness*).
- The *format* of the data meets the user requirements.

The nature of these aspects reflects the application area. Aeronautical data is very often concerned with physical measurements of such quantities as altitude or direction. Here accuracy and resolution are key factors in determining data quality and have a direct influence on safety. However in many cases data does not represent an analog measurement and these measures are inappropriate. If a data element represents a binary quantity or a printable character, the concepts of accuracy and resolution are meaningless.

When dealing with a data value that represents a digital quantity it may be appropriate to assume that the value is either correct or incorrect. However, when dealing with a data value that represents an analog reading, there will not generally be a uniquely correct value. All analog measurements are subject to errors and thus all such measurements will be subject to variability. Thus when defining the requirements of measurement data we need to define the required resolution and accuracy of the data.

It is clear that the quality requirements of data are not simply a case of defining whether we need 'high quality' or 'low quality' data. The meaning of quality varies considerably with the *nature* of the data.

Quality also varies with *time* since one element of quality identified in DO 200A relates to the *timeliness* of the data. A data item may be of high quality at the time it is created but may become invalid, or of lower quality, as time passes. For example, data describing the position of a train on a track will be valid for a relatively short period if the train is traveling at high speed.

Data integrity requirements

The integrity requirements of a system reflect the importance of the correct operation of that system. The generic standard IEC 61508 [ref. 2] describes the level of criticality of a system by assigning it a safety integrity level (SIL). It then goes on to describe appropriate characteristics and development methods for systems within each of these integrity levels. The standard also gives target failure rates for systems of each SIL and these are shown in Table 1.

Table 1. Target failure rates for systems of different safety integrity levels from IEC 61508

SIL	**High demand or continuous mode** (Probability of a dangerous failure per hour)	**Low demand mode** (Probability of failure on demand)
4	$\geq 10^{-9}$ to $< 10^{-8}$	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-8}$ to $< 10^{-7}$	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-7}$ to $< 10^{-6}$	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-6}$ to $< 10^{-5}$	$\geq 10^{-2}$ to $< 10^{-1}$

The safety integrity requirements of a system impose corresponding requirements on its component parts, and IEC 61508 defines 'hardware safety integrity' and 'systematic safety integrity' as influencing factors. The latter includes the effects of all systematic aspects and includes 'software safety integrity'. The standard does not specifically define 'data safety integrity' but following our treatment of data as a separate entity this seams a reasonable extension. Within this paper we refer to data safety integrity as *data integrity*.

While integrity requirements are more than just a set of target failure rates, these targets *are* of importance. In an arrangement consists of several component parts, the overall number of system failures will be equal to the sum of the system failures produced by each component. Therefore, if a system consists of hardware, software and data elements, the overall target failure rate may be apportioned to provide separate failure rates for each element. This implies that in a data-intensive system of a particular SIL, one aspect of the data integrity requirements should be that data errors should not produce system failures at a rate greater than that allocated to the data component. It also implies that the data will require a target failure rate that is *lower* than the figure given for the corresponding SIL in Table 1.

While it might seem obvious that data should not cause system failures at a rate greater than the target system failure rate, the survey suggests that this requirement is rarely explicitly applied to data within data-intensive systems. One reason for this (apart from the observed tendency to ignore data generally) may be the problems involved in demonstrating that such a requirement has been satisfied. It is generally accepted that it is not possible to demonstrate that a system meets these target failure rates by testing along (except perhaps for low SIL and probably small, simple systems) and so confidence must be gained from a combination of dynamic testing, static testing and evidence from the development process. Over the years the various safety-related industries have gained considerable experience in assessing the safety of software-driven systems. A wide range of static code analysis tools are available to investigate the properties of software, and standards such as IEC 61508 give detailed guidance on which development techniques are appropriate for systems of each SIL. Using these techniques, together with appropriate dynamic testing, it is possible to have reasonable confidence that the target failure rates have been achieved. However, the situation with regard to data-intensive systems is very different. Few static tools are available to investigate the correctness of data, and little guidance is available on the development techniques appropriate for such systems. Given this situation it is difficult to see how a system developer can reasonably demonstrate that their data-driven system will satisfy these requirements.

A further problem with data-intensive systems relates to the changing nature of data. While the hardware and software elements of a system are normally fairly consistent, the data is often continually or continuously changing. This means that the developer must be able to demonstrate not only that the system satisfies the appropriate target failure rate, but also that it will continue to do so with any appropriate set of data. If the developer is unable to do this, then presumably the system should be re-validated each time the data is changed (which would normally be completely impractical).

Data supply chains

The data used within data-intensive systems may be supplied to the system in a number of ways. In some cases the data may be developed specifically for the application during the development phase, while in others it may be produced at installation time to configure a system for a particular installation. Data may also be input directly by an operator while the system is off-line or in use. In many cases data-intensive systems form part of a hierarchy of systems where data is supplied by other systems. In this last situation data may be produced on one system and then be transmitted (perhaps being modified or reconfigured) by a range of machines before reaching its final destination. Where data-intensive systems are linked to a distributed information system, the same data may be used by a range of machines for very different purposes. Under these circumstances the requirements of the data (including the integrity requirements) will vary between these machines. For example, in a railway system, data that represents the current position of the trains is used by signaling control systems (where its use is safety-related) and also by passenger information systems (where it is not).

The fact that data may be produced separately from the target system leads to the concept of a *data supply chain* which includes all the elements involved in producing, communicating, preparing and formatting the data, before it is supplied to the system that will use it. RTCA DO 200A [ref. 8] describes *aeronautical data chains*, and provides much useful guidance in this area.

The existence of a chain of elements responsible for the supply of data greatly complicates the process of defining data requirements, and ensuring its quality and integrity. It also raises the issue of who is responsible for achieving the required integrity within the data?

Responsibilities for achieving data integrity

In large systems several elements, or participants, may be involved in the supply of data, and this may well involve a range of different organizations. Since the same data may be used for a range of purposes, it is essential that it satisfies the requirements of each application. Who then bears the responsibility for ensuring that this is the case?

RTCA DO 200A says:

> "*When the achievement of data quality depends upon the quality of data obtained from a previous participant, then either the data accepted from the previous participant must be validated to the required level, or an assurance of data quality must be sought from that pervious participant. For the majority of aeronautical data there is no benchmark against which the quality of data accepted from a previous link can be validated. The need to obtain assurance of the data quality will therefore normally flow back through the system until it reaches the originator of each data element. Consequently, reliance must be placed upon the use of appropriate procedures in every stage of the process.*"

This extract raises several interesting points. Firstly, it highlights the problems of validating data within an application – in many cases it is very difficult to determine the correctness of data without reference to its source. Secondly, it places the responsibility for ensuring data integrity on the end-user of the data. This would seem appropriate since it is the end-user who bears ultimate responsibility for the safety (and the functionality) of the system, and because only the end-user is in a position to determine whether the data meets the data requirements. In this case it is assumed that the end-user ensures data integrity by validation of the data supply chain. While it may seem logical for the responsibility for data integrity to rest with the end-user, as the supply chain becomes longer this responsibility becomes increasing difficult to exercise. Tillotson [ref. 9] suggests that in such cases the responsibility for data integrity may need to rest at the data source. He observes that where data entry occurs 3, 4 or perhaps 5 systems away from where it is used, the end-user's ability to ensure integrity becomes diminished. Under such circumstances, he suggests, responsibility must lie with the data originator. Unfortunately, the different uses of the data make this problematic.

Discussion

It is clear that many issues relating to the preparation and use of data are not being appropriate addressed. Data is increasingly being used in safety-related applications, and yet the various standards in this area say almost nothing about its requirements and development. In most standards data is assumed to be part of the software, yet the general guidance given on software does not address issues that are unique to data.

Perhaps the most worrying issue is that data is often not subjected to thorough hazard and risk analysis. This masks the significance of data to the safety of the system, and in turn reduces its perceived importance. In the absence of identified hazards the data is seen as unimportant, and appropriate safety requirements are not imposed.

Many of the problems associated with data may be tackled by identifying data as a distinct system element. This emphasises the importance of data and simplifies the task of preparing specific guidance. Giving data its own development lifecycle can ensure that it is subjected to appropriate hazard and risk analysis, and can enforce the assignment of data safety requirements. The integrity requirements applied to the data will then be used to determine appropriate development targets and methods.

Unfortunately, the assignment of requirements and targets does not solve *all* the problems associated with data. It is meaningless to assign a target failure rate to the data of a system unless we are in a position to demonstrate that this target has been achieved. For all but the lowest levels of safety integrity the target failure rates are too demanding to be demonstrated by dynamic testing alone and it is necessary to use other methods to gain confidence in the integrity of the system. When considering software we achieve this additional confidence through the use of an appropriate development process, including the use of dedicated static test tools. However, in the area of data we have little knowledge of the effectiveness of different development methods and few tools to aid our work.

Experience shows that data-intensive systems are often relatively complex. They often form part of a hierarchy of computers that interchange data of various forms. This interchange introduces multiple interfaces, each capable of introducing errors into the data that crosses it. Validation of such systems is extremely difficult. One approach would be to consider the hierarchy as a single entity and to attempt to validate the arrangement as a whole. This is generally impractical, partly because of the size of the resulting system, and partly because the hierarchy is likely to include large, corporate information systems that are not themselves safety-related and are not designed to permit validation. A more practical approach is to consider the safety-related sub-systems as self-contained entities and to validate them individually. However, where these are data-intensive systems this is still problematic. In most cases the safety of these systems will be dependent on the data that they receive from other machines within the hierarchy. Since some of these machines may have a relatively low level of integrity it may not be possible to rely on these external elements to produce adequate data integrity.

One approach to the validation of such data-intensive systems would be to partition them so that the data represents an effectively isolated partition. The system would then be validated to show that the system provides adequate safety and integrity *for any valid data set*. It would then be necessary to validate incoming data to ensure that it satisfies the appropriate safety and integrity requirements. However, despite the elegance of this approach it also presents problems. Firstly because it may be extremely difficult to prove that a system is safe for *any* valid data set, and secondly because validation of data is often very difficult. This latter point is illustrated by the quotation from DO 200A given earlier, which states that "*For the majority of aeronautical data there is no benchmark against which the quality of data accepted from a previous link can be validated*".

There is urgent need for additional work on the role of data in safety-related systems. This is needed to collect experience from a range of industries and to produce both generic and industry specific guidance on how data should treated. In particular, work is needed on the validation of data and data sources. There is also a need for additional tool support in this area.

Conclusion

We are increasingly seeing the use of data-intensive systems within safety-related applications. The safety of such systems is determined by all its constituent parts – including its data. Unfortunately, there is much evidence to suggest that data is not treated with the same care as other system components. In particular: data is often not subjected to hazard of risk analysis; data is generally not given specific safety or integrity requirements; and data is often not subjected to rigorous verification or validation. A possible reason for these omissions is that data is a largely ‘forgotten’ system element. It is not specifically addressed within the various standards in this area, and very little guidance is available within the literature.

One way of ‘raising the profile’ of data is to identify it as a distinct system component and to assign to it a dedicated development lifecycle. This would ensure that it was not forgotten during the development process and would simplify the task of providing appropriate guidance and requirements. Data should be subjected to appropriate hazard and risk analyses and any identified hazards should be reflected in the system requirements, including in the data safety requirements. Data should also be investigated to determine its impact on safety and appropriate data integrity requirements should be allocated.

Data requirements include consideration of data quality, which reflects its fitness for purpose. Where data items represent measurements or other analogue quantities, quality will include factors such as accuracy and resolution. However, for many digital quantities these terms are meaningless. Thus the nature of data quality reflects the form of the data items concerned.

Since data errors have the possibility of creating systems failures, requirements of *system* integrity will impose requirements of *data* integrity. IEC 61508 proposes target failure rates for systems of different safety integrity levels and these in turn imply limits to allowable failure rates as a result of data errors. Data integrity requirements will also determined the development methods that are appropriate for the data and may influence the choice of architecture used. Systems with high data integrity requirements may need to use data fault detection or data fault tolerance techniques to achieve the required system failure rates.

Data is often delivered to a data-intensive system through some form of data supply chain. In complex systems this chain may involve several computers and perhaps several organisations. Since the same data may be used for a range of purposes, it is essential that it satisfies the requirements of each application. It also raises the issue of who is responsible for achieving the required integrity within the data? Standards such as DO 200A assign this responsibility to the end-user of the data but in some cases it will be very difficult for the end-user to guarantee this integrity. Some workers consider that in some cases the responsibility for data integrity must lie with the originator of the data.

While the assignment of data safety requirements is a necessary step in achieving overall safety, it is also necessary to demonstrate that these requirements have been met. Unfortunately this is often very difficult. Data integrity requirements usually require the developer to demonstrate a system failure rate that is lower than can be demonstrated using testing alone, and at present we have few process-related measures that increase our confidence in our data development methods.

Data-intensive systems often form part of a large hierarchy of computers that often includes corporate information systems as well as safety-related systems. Data is interchanged between these computers and may be used for different purposes within different machines. This complexity causes tremendous problems for the verification and validation of any safety-related systems within this hierarchy. At present little guidance is available on appropriate methods of tackling this problem, but this paper suggests that system designers should attempt to partition their systems such that data represents an effectively isolated partition. Attempts should then be made to show that the system is safe for any valid set of data, and to validate all data before, or as, it enters the system.

Many problems exist in the use of data-intensive systems within safety-related applications and much work is needed in this area.

References

1. Neil Storey and Alastair Faulkner "The Role of Data in Safety-Related Systems", *Proc. 19th International System Safety Conference*, (Huntsville 2001).

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

7. Neil Storey and Alastair Faulkner, "Data Management in Safety-Related Systems", *Proc. 20th International System Safety Conference*, (Denver 2002).

8. RTCA: DO 200A *Standards for Processing Aeronautical Data*, (Washington: Radio Technical Commission for Aeronautics, 1998).

9. Tillotson J., "System Safety and Management Information Systems", *Aspects of Safety Management: Proceedings of the 9th Safety Critical Systems Symposium*, (Bristol, 2001).

Biographies

Neil 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, email - 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 Expert Panel on Safety-Critical Systems and has a large number of publications including both journal and conference papers. Neil is 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. Alastair's research interests are in the safety 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.