# The importance of data in safety-critical systems

**Neil Storey**

**Introduction**

All computer-based systems make use of data in one form or another and it is common to consider this data as an integral part of the system's software. However, an increasing number of systems make use of data to configure the system or to describe its environment, and in such cases data often forms a distinct element. Data in these 'data-driven' systems is often generated and maintained quite independently from the executable software, and there is some evidence to suggest that in many cases it is not receiving the attention it deserves.

The high development costs of safety-related systems encourage the use (and re-use) of standardised hardware and software wherever possible and this 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 [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. Other examples of data-driven applications include those that use data to model a physical or operating environment. These include applications such as air traffic control (ATC) and railway management systems.

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 [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 such as those in the civil aircraft, military, nuclear and railway sectors [3-6] give very little guidance in this area. One of the few industrial sectors that do have standards relating to data is the air traffic control sector, which has created the DO 200A family of standards in connection with the processing of aeronautical data [7,8]. These standards contain much useful information, but are highly industry specific and are not widely used (or known of) in other sectors. They also concentrate on the *processing* of data, and say little about its design, production or verification.

Perhaps one of the reasons why these documents give so little prominence to data is that they make the tacit assumption that it 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, the techniques required in the production and maintenance of data are often quite different from those used for *executable* software. It is also clear that in many cases configuration data is *not* developed alongside more conventional software, and is not developed with the same degree of care.

**Data Development**

Since data forms a significant and important element within data-driven systems, one would expect that the development methods used to produce it 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. An informal survey of a range of industrial sectors [9] suggests that data is often:

- Not subjected to any systematic hazard or risk analysis.
- Not given any specific safety requirements.
- Not assigned any specific integrity requirements.
- Poorly structured, making errors more likely and harder to detect.
- Not subjected to any form of verification.
- Drawn from a single source.

Many of the problems identified within the survey follow from the first point listed above. Data is often ignored within the process of hazard and risk analysis, possibly 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, suggesting that the data has no specific safety requirements. This is perhaps the reason why no specific safety integrity requirements are normally assigned to the data. Experience shows that 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 usually has no integrity requirements, such techniques would not normally be considered.

Since data often has no specific safety requirements and no safety 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.

**The changing nature of data**

One of the characteristics that differentiate data from the executable elements of software is its tendency to change. Some data is inherently dynamic and is constantly changing. Examples of such data include that used to describe the current positions of aircraft in an ATC system and the positions of trains within a railway network. Many data-driven systems also use slowly changing (or pseudo-static) data to describe their environment. Using the same two examples, this form of data would include that used to describe the physical layout of the airspace of an ATC system (for example, the positions and heights of airfields and mountains) and the layout of the track and points of a railway. It is tempting to think that such data is static (non-changing), but this is invariably not the case. For example, geographical changes (such as a change in the design of the airways) dictate that aeronautical data is updated every 28 days.

In addition to dynamic and slowly changing data, most systems will also include configuration data that is normally assumed to be static and unchanging. However, in many cases, one of the key reasons for adopting a data-driven implementation is the ability to make rapid changes to the configuration simply by modifying the data. Thus the configuration data may be regularly changed to adapt the system to changes in plant or required functionality.

**The implications of data change**

When a safety-related system is modified by making changes to its *software*, it would be normal to revalidate the system to ensure that the changes have not affected its safety. This would suggest that when using a data-driven system, the system should be revalidated whenever changes are made to the *data* being used. In reality this is impractical since some elements of the data are changing continually. This would seem to suggest that the original validation of the system should prove that the system is safe for *any* combination of data. While this might be a utopian solution, in most cases such validation would be impossible.

An alternative and less demanding approach might be to design the system such that the data represented an effectively isolated module within it. It might then be possible to define some specific safety requirements for the data, and to validate the system for *any data set that satisfies these requirements*. This approach (if it could be achieved) would mean that changes to the data would only require the validation of this data, rather than re-validation of the complete system.

In fact, in many cases neither of the above approaches is adopted. No attempt is made to partition the system to allow the data to be considered as an isolated module, and data is simply changed without any attempt to validate the data or to re-validate the overall system. This approach would seem to make the tacit assumption that the system was originally validated for any set of data, without any attempt having been made to achieve this in practice.

This leaves us with the problem of how we should (or could) validate systems with changing data.

**The validation of data-driven systems**

The key to validating data-driven systems lies in designing such systems to permit and simplify validation. Design for testability is not a new concept, but in many cases this principle is not being applied to data. The design of data structures must be undertaken with the same care as that of software

and hardware components, and similar considerations of partitioning, isolation and, if appropriate, fault tolerance, should be applied. Having said this, it is clear that we do not yet have recognised techniques for dealing with the unique problems associated with validating data-driven systems, and much more work is required in this area.

A basic pre-requisite for system validation is the existence of appropriate requirements against which to judge the system. In data-driven systems it is essential that these include the safety and integrity requirements of the data, as well as those of other system components.

**Taking data seriously**

One way of encouraging developers to give data the attention is deserves might be to treat it as a distinct system entity with its own requirements and lifecycle. Defining 'data safety requirements' and 'data integrity requirements' as specific 'deliverables' within the development process would ensure that data is considered in the hazard analysis phase, and would ensure that data-related requirements are carried forward to later development phases.

It should be noted that the author is *not* proposing that all safety-related systems should treat data as a distinct element, or that the data used within conventional software should be treated separately. However, there is a strong case for considering data separately in data-driven systems, since the data is often developed quite independently from other parts of the system.

**Recognition of the importance of data**

While it is clear that data is currently not receiving the attention it deserves, there is mounting evidence to suggest that some industries (for example the railway and air traffic control industries) are beginning to recognise the importance of this system component. It also seems likely that the next revision of IEC 61508 will incorporate material specifically related to data integrity and to data management techniques. These developments should be applauded.

**References**


1. J. A. McDermid, The cost of COTS, *IEE Colloquium - COTS and Safety critical systems* London, 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, MoD, 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. RTCA: DO 200A *Standards for Processing Aeronautical Data,* Washington: Radio Technical Commission for Aeronautics, 1998.
8. RTCA: DO 201A *Standards for Aeronautical Information*, Washington: Radio Technical Commission for Aeronautics, 2000.
9. N. Storey and A. Faulkner, Data Management in Safety-Related Systems, *Proc. 20th International System Safety Conference*, Denver, 2002.