Data-driven systems – the state of the ark?

Neil Storey

Introduction

In the early days of safety-critical systems, engineers were reluctant to place their trust in anything more complicated than electromechanical relays and switches. Early computer-based safety-critical systems were often based on single-board or sometimes single-chip computers, and complex components such as operating systems, or complex techniques such as task scheduling, where avoided whenever possible. However, as computers became more powerful, and more prevalent in non-critical applications, the complexity of critical computer-based systems increased steadily. Inevitably, this increase in complexity was accompanied by an increasing use of COTS products to implement both the hardware and the software of the system.

Almost all computer-based systems make use of some COTS elements, since the microprocessor itself is invariably a COTS part. It is well known that the use of such components brings with it associated problems of verification and validation, since one's knowledge of the element is always limited. For this reason, in many highly critical applications, while components such as processors will be bought-in, much of the remaining hardware and software will be developed in-house. However, in less demanding applications, commercial pressures mean that inevitably much of the system will be constructed from mass produced parts.

It is interesting to note that the original development of the microprocessor was motivated by a need to allow the mass production of VLSI components. Electronic technology had developed to a stage where very complex components could be mass-produced with a very low unit cost. However, the *development* cost of the components was very high, making it only attractive to produce components in very large quantities. The development of the microprocessor allowed a single component to be produced by the million, to be customised for a wide range of applications by the use of software.

More recently, we have seen a similar process taking place with the increased use of data-driven systems. Here common hardware and common software can be used in a wide range of situations, by customising it for a particular environment using data. In some cases the data takes the form of a simple data table, which configures the device for a particular application. In others, the data might be a huge database representing the 'knowledge' required of this particular system. Data *can* be used to drive systems that are largely custom designed, but in many applications areas we are seeing an increasing use of COTS elements that are customised using data. In many situations this offers a very cost-effective implementation strategy.

When configuring a microprocessor for a given application by the addition of software, it is clear that the characteristics, and therefore the safety, of the resultant system are determined by the nature of the software as well as that of the processor. One would therefore not attempt to argue that a computer system was safe simply because the processor has been used in other applications that have been shown to be safe. Similarly, when configuring a data-driven system by the addition of data it would seem obvious that the safety of the system is determined partly by the nature of the *data*, and that experience gained using systems configured with one set of data is not sufficient to deduce that the system will be safe using any other data set. However, despite the seemly obvious truth of this conclusion, experience shows that many system developers are placing great faith in tests performed using a single set of test data during system development, and inferring from this a great deal about the overall performance of the system.

The increasing use of data intensive applications

The increasing use of data-dependent systems is not restricted to the area of safety. A vast array of IT and infrastructure projects are critically dependent on data – often in huge quantities. For example, the proposed UK identification card project is dependent on the ability to store and retrieve personal information for many millions of individuals, while the London congestion charging system must store data on cars, their movements and payments. A common characteristic of such systems is that they often link, and extract data from, a number of discrete data sources (or databases). For example, a police computer system might interrogate databases related to criminal records, boarder crossings, vehicle licensing, tax registration, medical records and other information sources. Experience shows that trends within 'commercial' applications inevitably go on to influence safety-related applications, and the current trend is certainly towards the use of larger and larger collections of disparate data sources to 'inform' system operation. This would suggest that we might expect to see further dependence on the use of large databases, and networks of databases, within safety-related applications.

Since it seems likely that data intensive systems will form an increasingly important part of safety-related applications, it would seem sensible to see how large data-driven systems are being managed within other sectors. What is clear, is that the IT sector has not solved all the problems associated with data! Topics such as 'Data Migration' and 'Information Risk Management' form the basis of many working groups and consultancy projects, and are generally seen as major problems in such applications. Many of the issues that cause concerns are related to issues of 'data quality', although this term is often poorly defined. A major problem is that IT professionals seem very reluctant to use quantitative measures to define requirements or performance. This inevitably leads to major problems in interpreting specifications and contracts.

Quantitative measures of requirements and performance

In terms of specifying the 'quality' required of a system, the IT sector could gain a great deal by looking at the safety sector. For some 20 years the norm within safety applications has been to represent the importance of the correct operation of a system by assigning it a safety integrity level (SIL). IEC 61508 then gives guidance on appropriate worst-case failure rates for each level of integrity. Over the years we have also gained experience in how to develop hardware and software to achieve these different levels of integrity.

Clearly, it would not be appropriate to assign a *safety* integrity level to an IT application, but it would seem appropriate to quantify the integrity required. For example, it would seem appropriate to define whether bidders for the contract to develop the national ID card system are required to produce a system with an error rate of one in a thousand, one in a million or one in a billion (where the nature of the errors is appropriately defined). Since the implementation costs increase dramatically with the integrity requirements, any choice of supplier based primarily on price will tend inevitably to favour a low integrity implementation, which could well explain the failure of many major IT projects in recent years.

Assigning numerical integrity requirements to major IT projects would spell out the required performance of the overall system and would inevitably place requirements on the correctness of all elements of the system. This would require the developers to look at all sources of errors, including not only the hardware and software of the system, but also problems produced by human error or incorrect data. Again, the IT industry could benefit from experience gained over the last few decades within the safety sector to help it to select appropriate hardware and software architectures and development methods to achieve particular levels of performance.

Achieving high integrity in data intensive applications

Before we in the safety sector become too self-congratulatory, we should perhaps note that although standards such as IEC 61508 give us excellent guidance on how to achieve appropriate levels of integrity in a range of applications, there is much evidence to suggest that many of us are *not* following this guidance when it comes to the development of data-driven systems. This omission comes about because many system developers are simply not seeing data as a distinctive element within their systems and are thus not treating it with the care it deserves and requires.

Back in 2004 I wrote an article for this Newsletter in which I outlined the results of an informal survey that looked at the development of data within a range of industrial sectors. This survey suggested 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.

While it is several years since this survey was performed, there is much anecdotal evidence to suggest that this situation has not changed.

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.

By ignoring a significant element within the system, many developers are failing to apply the rigours of IEC 61508 to their system. However, the reason for this oversight is perhaps understandable, since IEC 61508, along with almost all others safety standards in current use, says almost nothing specifically about the creation, testing or use of data.

In the first section of this article I pointed out the obvious folly of concluding that because a system appears to work correctly when used with one set of data, it will be safe using any other data set. However, experience shows that this assumption is being widely made in the development of data-driven systems. Often the reason for adopting a data-driven approach is because a particular system is to be used in a range of similar, but not identical, situations. The first implementation is produced by a development team and is normally thoroughly tested using data appropriate to that system. Further installations are then produced by giving the system to an installation engineer who modifies the data to customise the system to its new environment. This installation engineer normally has limited knowledge of the design of the system or of the development tests used to validate it. Thus the testing performed is limited to some form of acceptance testing. This approach makes the tacit assumption that changes to the data will not affect overall safety – an assumption that is completely unjustified.

Since it seems likely that data-driven and data-intensive systems will become more prevalent and more complex, the problems associated with their development and use are likely to become more apparent.

Taking data seriously

A major problem in managing data within safety-critical systems is that, unlike in the development of hardware and software, there is no established 'best-practice' in this area and no meaningful guidance within the various standards. However, the most important step in tackling this problem is simply to identify data as a system component and to look at its effects on the overall system. If the effects of data errors are considered (through a process of hazard analysis) then a range of standard techniques can be applied to tackle them.

I would like to thank my colleague Alastair Faulkner of EamonGrace Ltd, for his suggestions and comments on this article.

Neil Storey is at the University of Warwick and can be contacted at N.Storey@warwick.ac.uk