The Safety Management of Data-driven Safety-Related Systems

A. G. Faulkner, P. A. Bennett, R. H. Pierce, I. H. A. Johnston

CSE International Ltd, Glanford House, Bellwin Drive, Flixborough DN15 8SN, UK
Tel: +44 1724 862169
Email:agf@cse-euro.com

N. Storey

School of Engineering, University of Warwick, Coventry. CV4 7AL UK

Abstract.

Many safety-related systems are built from generic software which is customised to work in a particular situation by static configuration data. Examples of such systems are railway interlockings and air traffic control systems. While there is now considerable experience and guidance on how to develop safety-related software, and there are a number of standards in this area, the topic of safety-related configuration data is hardly mentioned in the literature. This paper discusses the desirable properties of safety-related data and sets out principles for the safety management of such data, including a data lifecycle which is analogous to a software development lifecycle. Validation and verification of the data, and the means used to achieve such validation and verification are given particular attention.

Introduction

Many Commercial Off The Shelf (COTS) computer systems are constructed from a generic software platform which implements the necessary algorithms for the particular class of application, and is customised for a specific application. Common sense and good practice often dictates the use of configuration data in one form or another, rather than changes to the software code itself.

Examples of such data-driven systems include:

1. Railway interlockings, where the signalling control tables for a particular track layout are represented in the form of data;
2. Railway signalling control systems where data is used to represent the track layout and the location of each signal, points and other trackside devices;

3. Air traffic control systems (such as radar displays and Flight Data Processing (FDP) functions) are also configured by extensive data (also known as adaptation data) which describe the layout of airways and control areas, the location of navigational aids and many other aspects of airspace design; and
4. Telecommunication switches (exchanges) are also generic platforms that are customised to the needs of a particular network by data that provide the rules for call routing and associates telephone numbers with physical or logical circuits.

In these examples, the configuration data are often geographical in nature since the generic system needs to be adapted to the particular instances of a real physical environment.

The correct functioning of these data-driven systems is dependent on the correctness of the configuration data, just as it is dependent on the correctness of the software code. Errors in the configuration data can lead, in safety-related systems, to hazards and possibly to accidents.

Configuration data should be clearly distinguished from normal dynamic data maintained by the computer system. Configuration data represents the static model of the real world; dynamic data represents the current state of the model (as derived from sensors and human inputs). Configuration data input is part of the off-line system design and construction process, and should not be alterable in normal system operation. The safety management of configuration data is a topic that relates to both commercially available products (COTS) and specific applications developed in-house.

The issues of safety management of data

The desire amongst suppliers and operators to use COTS products [1] in safety-related systems for reasons of cost and timescale [2] raises a requirement for the safety management of data-driven safety-related systems.

McDermid [2] identifies that the construction cost of a system may be reduced by the purchase of COTS system. This COTS system will often consist of hardware and software elements, which will typically be tailored to the input domain by configuration data.

McDermid [2] further adds that the certification cost of this generic hardware and software may exceed the savings made by purchasing COTS products. This certification cost is incurred because the evidence required for approval of the COTS product may not be available to the buyer or may be inappropriate to the application domain.

The certification (or approval) cost of a data-driven system may be considered to be in two distinct parts:

1. That for the evidence for generic part of the safety-related system; and
2. That for the evidence for configuration data used to drive the safety-related system.

The generic part of a COTS solution (both the hardware and software) can be developed under existing standards such as IEC 61508 [3] using established tools and

techniques as a means of achieving the necessary certification or approval. This is not to say that the achievement of such certification is necessarily straightforward particularly for a generic COTS solution. It is usually easier to achieve certification for a system developed for a particular application, which does not include the uncertainties associated with possible other uses.

Examples of high integrity COTS systems for specific applications are found in railway interlocking systems. Here generic hardware and software components have been developed to a high level of integrity and granted type certification by railway regulators. In such cases the problem of overall system validation is, in principle, reduced to that of ensuring the integrity of the configuration data.

The development of the hardware and software parts of a generic solution is extensively supported by safety-related literature in broadly the same way as the development of a custom solution for a particular application. There are established processes, and detailed guidance, for hardware and software that cover a wide range of equipment and systems. These range from application specific hardware based protection systems implemented with little of no software content, to complex monitoring and control software systems based upon standard computing hardware.

For the software element of such systems there are now development languages and translation tools specifically aimed at safety-related systems. The desirable characteristics of the software, such as code structure, are well understood, even if that understanding is sometimes patchily applied. The use of tools to measure the static and dynamic properties of the software is becoming common.

The configuration data that drives a COTS system in a particular instance is itself, in most cases, not COTS (although a counter-example is given later). This application specific part of a COTS solution is likely to be created by the system integrator, who in many cases will not be the COTS product supplier. It may be treated as a one-off application developed specifically for a particular instance of the system.

The issues raised by the use of configuration data in data-driven safety-related systems are not directly addressed in the safety community literature. The need for data integrity commensurate with the SIL of the software is recognised in some standards such as IEC 61508 [3], Def Stan 00-55 [4], and CENELEC prEN 50128 [5]. However, these standards provide very little guidance on how data integrity is to be achieved, although prEN 50128 [5] is the most useful in this respect and at least recognises the need for a defined data lifecycle**.** As a result there is no established guidance on the process for generating such data, or on methods of achieving certification for it.

The problems of configuration data are most acutely seen where there are large volumes of real-world configuration data to be captured. Many systems need some form of configuration data, but if this is confined to a few scalar parameter values then the effort in validating these values will be small and the risk associated with this data correspondingly small. The real concern arises where large volumes of data are required.

The remainder of this paper sets out to identify the important issues and to indicate how these issues might be addressed either by using existing techniques or by developing new ones. The central issues are those of validation and verification, process evidence, data structure, tools and techniques. There is also a need for an

agreed set of requirements for configuration data for a range of integrity levels, for example SIL1 to SIL4 in IEC 61508 [3].

Hazard analysis for configuration data

The safety requirements of any system are identified through a process of hazard and risk analysis. Hazard analysis should consider the problems produced by incorrect data alongside problems associated with faults within hardware and software. In this way an integrity requirement should be assigned to the configuration data of a system, as for other system components. Standards such as IEC 61508 [3] give guidance on techniques to investigate hazards associated with hardware and software, but say little about the data within systems.

Hazard analysis involves the investigation of the functionality provided by the system and the consequences of the failure and failure modes of the identified functionality.

A variant of the Failure Modes and Effects Analysis (FMEA) method would be a possible basis for such an analysis. In the case of data the FMEA should consider both errors in data types and where necessary in individual data entities or values.

As an example, consider a flight data processing (FDP) system. The purpose of such systems is to distribute data on the proposed flight plans of aircraft to allow the air traffic controller to be aware of the presence and intentions of aircraft in his or her sector of airspace or shortly to enter it. Such information aids the controller in maintaining the mental "picture" of the air traffic and how to control it, and also acts an important reversionary measure in case of loss of radar surveillance data. Flight data is presented in the form of flight (progress) strips, either printed or on electronic displays. Typically an FDP system will generate a flight strip for a given sector some time in advance of the time a flight is expected to enter the sector. Since airspace design changes from time to time, it is usual to store sector boundaries as configuration data. The data failure mode "sector boundary in wrong position" may cause the system to fail to produce a flight strip for a given flight, or direct it to the wrong sector, which would lead in turn to a hazardous system failure mode "flight strip not produced when required". This kind of error would become more critical if the "free flight" concept becomes widely used (airliners being given direct routing from point to point rather than following linear airways) in which case flights could enter sectors at many different places.

The above example shows an FMEA analysis of a data *type*, since there are many sector boundaries in a typical FDP system.

An FMEA analysis applied to individual data *elements* might consider the trip threshold values entered for individual sensors in reactor or plant protection systems. A wrong data value might delay a trip until a hazardous plant state had been reached. Each element value would have to be considered individually since some may be more critical than others.

Systems such as railway interlockings and telecommunications systems make extensive use of configuration data. Railway interlockings are a specific example of a safety-critical system that clearly illustrates the use of configuration data.

In the example of the railway interlocking the generic part of the system is customised to the application instance by configuration data. A hazard analysis of the railway interlocking should identify not only the consequence of failure of the generic hardware or software, but also the consequences of inaccurate, erroneous or corrupted configuration data.

Configuration data failures may range from gross errors due to random failures in hardware, to systematic errors in the data preparation. Systematic errors in data preparation are due to failures of the data generation (collection) and management, or as a consequence of changes to the real world which that data represents. Changes to the real world may be as a consequence of a maintainer exchanging equipments for different makes, models or issues of the existing equipment.

Validation and verification

Validation and verification of configuration data is central to the safe operation of any data-driven safety-related system. The definition of the configuration data and the closeness of its representation to the real world, which it aspires to describe, will be key factors in maintenance of the configuration data once the safety-related system enters service.

A system that uses terrain data to guide an aircraft between two points, or in a ground proximity-warning device, may be taken as an example.

What risk would be posed by terrain data which contained errors? In the case of a guidance system for low flying, it could lead to 'negative ground clearance' and the loss of the aircraft. In this case, the severity of the hazard is high and the integrity of the data must be correspondingly high to achieve a tolerable accident rate. If the terrain data is being used in a ground proximity warning system (GPWS), there has to be an error on the part of the pilot before the GPWS is needed, and although the severity of the resulting accident is the same, the likelihood of it is reduced and a lower integrity can be tolerated for the data.

The degree of rigour used in validation of configuration data should therefore be commensurate with the risk (hazard severity and likelihood) posed by errors in the configuration data and the degree to which the safety-related system could tolerate errors in the data.

What measures, tools and techniques could be used to reduce errors in the terrain data to an acceptable level? One possibility is to consider how the data is represented within the computer.

In the above example, the terrain data could be represented as a collection of heights above sea level on a grid representing the surface of the Earth. Such an array of values contains no inherent structure with which to perform on-line or off-line validation. Rules could be devised to identify the maximum height of a mountain, and the lowest part of a valley, within particular geographical areas. These minimum and maximum values could then provide range checking to detect gross errors. However, plausible error values could exist between the identified minimum and maximum values. Wire frame computer models, which consist of many triangles to describe the surface of the terrain, offer the opportunity to introduce further diversity

and self-checking capability into the representation of the terrain. Any single point within the terrain would be represented as the corner of a number of triangles. The relationship between each of the constituent triangles and the point which makes up a corner of these constituent triangles provide an inherent structure which can assist in the validation the terrain. For example, missing triangles (apparent "holes" in the terrain surface) or non-contiguous areas may be detected, indicating some corruption in the data.

The representation of terrain as a wire frame model will require more computing power than a simple grid model, so there may be a trade off between speed of computation and the ability of the data to reveal errors.

Another problem with geographical data, which is common to all representations, is that changes in the real terrain, such as the erection of a radio mast, must be reflected in the data. This is a particularly difficult problem in cases where real world changes are not within the control of the data user or originator. Geographical data is an example of data which may itself be COTS in that it is typically produced by a mapping agency rather than by the user of the application system.

Process evidence

Configuration data for safety-related systems should be created and managed under some systematic set of processes. The definition and design of the data structures, and of tools to manipulate and maintain the data content, should be derived in some systematic manner. This set of systematic processes may be collectively termed a data lifecycle. A lifecycle does not include evidence, but may include provision for generating process-based evidence.

Errors in configuration data used for data-driven safety-related systems will, as noted above, lead to risk. Established safety management tools and techniques should be used to analyse the risk presented by errors in configuration data.

Process evidence from the construction lifecycle for the generic part of the system will be derived from the hardware and software components of the system. The construction lifecycle should also include design evidence for the configuration data.

The population of the configuration data for a specific application instance of the data-driven safety-related system should also provide evidence for assurance purposes. The maintenance and management of the configuration data must continue over the life of the system.

The means of data preparation should be such as to minimise the likelihood of errors being introduced and maximise the likelihood of errors being detected. Data entry methods and formats should be as close to the real world representation as possible. For example, when entering data from a map, it is preferable to use a digitiser than to read off the coordinates of points of interest by eye and enter them by keyboard. This is an obvious example, but the principle is one which should be applied more generally.

The configuration data lifecycle should include:

1. The definition of the internal structure of the data, and its external representation;
2. The data validation techniques to be used;

3. A suite of test datasets;
4. Software tools to manage the configuration data (including translation between internal and external data representation); and
5. The definition of processes and procedures to manage the configuration data and the process evidence required for assurance purposes.

When testing the safety-related system, consideration should be given to coverage measurements of the data. In other words, it should be known what proportion of the data elements have been accessed by the software. Ideally 100% of the data elements should be accessed during system test, but whether this is possible, practicable or even sensible depends on the nature of the data and the application.

Software development should generate a number of test harnesses by which functional groupings of code may be exercised under the controlled conditions of a test specification. A corresponding structure for testing configuration data would be a test data set.

A suite of test data sets should be used to exercise the data-driven system under predetermined test conditions, these test conditions being described by each data set. Each data set should represent operation in either a normal or degraded mode with specified input conditions such as alarms and a description of the expected response from the safety-related system. The data-driven safety-related system should provide an opportunity to periodically test the system either for assurance of the systems continued safe operation or for fault determination. The data-driven system offers an opportunity to validate the system in the operational domain. The test data sets should be suitably separated from the operational domain so as not to become confused with the normal operation of the system.

Configuration data will probably require a set of software tools to manage and manipulate both the configuration data and the test data sets. These software tools should facilitate the offline validation of the configuration data.

The definition of processes and procedures for the configuration data recognises that people will be involved in the installation or operation of the safety-related system, or both. These processes are intended to provide a systematic framework, to be refined through experience and use, to reduce systematic errors in the configuration data.

Configuration data structure

Design features of the configuration data should be selected which reduce the likelihood that hazardous errors will remain in the data when the system goes into service and ease the task of data validation.

The data structure of the configuration data should be such as to enable demonstration that there is sufficient modularity to reduce side effects when the configuration data is maintained or extended. The configuration data should be complete, and it should not contain elements or structures which are not required. Where the configuration data has other features or structures then evidence should be provided that these features or structures are independent of the requirements of the application instance. Data elements (data content), which are not required for the

application instance, should be removed from the configuration data, unless it is not reasonably practicable to do this.

Validation and verification should be a prime design consideration for configuration data. Design options which ease, or make feasible, online or offline validation of configuration data should be chosen in preference to those design decisions which simply ease the design of other elements of the system such as software, hardware or fit with a current data set. Where possible the design option for the data structure of the configuration data should be requirement driven, and solution independent. The current data set may not lend itself to the validation requirements and hence be unsuitable for use in the safety-related data-driven system.

Tools and techniques

What tools and techniques are appropriate to configuration data used for safety-related systems? The tools and techniques used for data preparation and data maintenance should be appropriate to the risks posed by the use of configuration data in a data-driven safety-related system.

Current standards such as IEC 61508 [3] identify development, verification and validation and techniques for software, which are recommended, based upon the required safety integrity level (SIL1, SIL2, SIL3 and SIL4), with more rigorous techniques being used for higher integrity levels.

The definition of what tools and techniques are necessary for configuration data for a particular safety integrity level requires further debate amongst the safety community as a whole.

Examples required could include the following:

1. Techniques for data validation and verification could include inspection of the data by individuals independent of those who originally collected and entered the data;
2. Where the data is translated from an external human readable representation to an internal representation (which is normal), measures should be devised to check that the translation process does not corrupt the data, for example by reverse translation and checking against the original data;
3. The data entry format should be such as to minimise the chances of error. For example, where map data is to be entered, it would make sense to use a drawing tool rather than lists of co-ordinates of lines and points in numeric form (there is at least one FDP system which requires the latter, and despite internal validation checks in the data translator the system in question has failed in operation due to errors in the data); and
4. System testing will clearly be an important means of validating the data together with the software design and implementation. However, tests should be designed specifically to check that the software has accessed each data item. In the example of the FDP system given above, simulation data could be used to ensure that a flight strip is produced at the correct time when a flight crosses every sector boundary in the airspace.

Requirements for configuration data used for safety-related systems

The design of the configuration data should facilitate the validation of the data content and structure as appropriate to the risk presented by errors in the data set. If safety analysis exposes a high risk based upon failure of the configuration data content or structure, then the system requirements should include online validation (dynamic checking of the data).

The representation and means used for configuration data preparation should be as intuitive, convenient, and close to conventional real-world representations as possible, to minimise the likelihood of the data preparation process introducing errors.

Care must be taken in collecting the source information from which the configuration data will be created, to ensure that it is an adequately correct representation of "ground truth" (the source information may include geographical map data, engineering drawings, photographs, plant specifications and even human memories). This can be a particularly time consuming exercise if the source information is known to be out of date or unreliable, or there are doubts about its accuracy. New geographical or engineering surveys may be required.

A configuration data definition should support the concept of modularity, where this is logically feasible. A module of data should be self-consistent, complete, and contain no unresolved internal references, to facilitate automated rule-based validation.

Where references to other data items and data structures are made within a configuration data module these references should be made in such a way as to facilitate their validation. Data references (keys) should be based upon compound data structures. These compound data keys contain sufficient information to identify the referenced item without any addition external data. These compound keys would be in contrast to integer based keys, which would only allow range-based validation checks.

Conclusion

It has been argued above that there is a need for the safety management of configuration data, and a number of principles have been outlined. These principles are in addition to the requirements or guidance found in a number of standards, which is in most cases very weak.

More work is required in the area of data driven safety related system to establish clear guidance as to the design and management of data driven systems. The assurance of the configuration data should be supported by process evidence from an identifiable lifecycle for the system as a whole that should include the configuration data.

The population of the configuration data should be supported by process and procedures to reduce systematic error. These processes and procedures should not only be capable of the reduction of systematic error in the creation of configuration data, but also be used for the management of the configuration data through the

system life. These processes and procedures should not only be concerned with control of the introduction of errors but also the detection of errors.

The design of the configuration data should aide and facilitate the validation and verification of the data through data structures that lend themselves to rule based automation. The configuration data should support the use of 'data sets' that may be used to calibrate, test, and exercise the entire system. These data sets are created as separate modules to fully exercise the system in all functional conditions allowing demonstration of normal and degraded modes. These data sets would allow the system administrator to either detect faults in the system operation or to gain confidence in the continued correct operation of the system.

References

1. R. H. Pierce, S. P. Wilson, J. A. McDermid, L. Beus-Dukic and A. Eaton, "Requirements for the use of COTS operating systems in safety-related air traffic services", Proceedings of Data Systems in Aerospace, Lisbon, Portugal, 17 May 1999
2. J. A. McDermid "The cost of COTS", IEE Colloquium - COTS and Safety critical systems. January 1998
3. International Electrotechnical Commission, Functional Safety: Safety-related Systems, International Standard IEC 61508, January 2000.
4. UK Ministry of Defence (MoD) Def Stan 00-55: Requirements for Safety Related Software in Defence Equipment Issue 2 – Dated 1st August 1997
5. European Committee for Electrotechnical Standardisation CENELEC prEV 50128: Railway Applications: Software for Railway Control and Protection Systems Final DRAFT June 1997.