Improving the Analysis of Data in Safety-Related Systems

Report on a Project Submitted in Partial Fulfilment of the Requirements for the

Postgraduate Diploma in Safety Critical Systems Engineering

Presented to the Department of Computer Science

of the University of York by

James Inge

Supervisor: David Pumfrey

Number of words = 27141, number of pages = 58, as indicated by the Microsoft Office Word 2003 "word count" tool, including the title page, preliminary pages, report body and appendix.

1. Abstract

The behaviour of many complex systems is based not only on their static design but also on configurable data used by the system. In order to assess the safety of such a system, it is necessary to have an understanding about the types of problem such use of data may cause, through a process of safety analysis. This report reviews current guidance and best practice for treatment of data in safety analysis. It finds that while advice exists on how to manage the safety impact of data use after a potential hazard has been identified, there is a lack of tools and guidance for the initial task of hazard identification for safety-related data.

A taxonomy of types of data fault is proposed that can be used as a checklist to aid in the hazard identification process. The taxonomy is then validated using accident investigation reports, to determine whether it is capable of classifying the data-related issues seen in real life.

2. Preliminaries

2.1. Table of Contents

2.2. Table of Tables

2.3. Table of Figures

2.4. Declaration

All work presented in this report is that of the author, except where the work of others is explicitly acknowledged. In particular it should be noted that views expressed in this report are those of the author and do not necessarily represent the view of his employer, the Ministry of Defence.

2.5. Acknowledgements

Completing the SCSE course and this project in particular has taken an inordinate amount of my spare time and I must thank my ever understanding fiancée, Helen, for taking care of me while I worked and putting up with my absences.

I must also thank my supervisor, David Pumfrey, and the rest of the High Integrity Systems Engineering group at the University of York Department of Computer Science for their excellent tuition and continued support, and for making the whole course a thought-provoking experience.

Final thanks must go to my employers, the Ministry of Defence, for paying for my way and allowing me the time to attend the taught modules of my course.

2.6. Statement of Ethics

This project does not involve or promote any harmful activity. It is intended to advance the discipline of safety management in a way that will help avoid harmful accidents. It has been conducted as a desktop exercise that has not required the participation of others. The accident data used in this report has been taken from public sources and does not identify individuals.

2.7. Motivation

"The more data we have, the more likely we are to drown in it".

Nassim Nicholas Taleb [49]

A large amount of effort has been spent investigating the relationship between the design of systems (both hardware and software), and the unintended harm that might arise from their use. Traditional system safety has been concerned with ensuring that these systems are correctly designed for safe function, that the designs meet their users' true requirements, and that they will continue to operate correctly, or in the case of failure, fail in a known safe manner.

With the increased use of ever more powerful data processing hardware and configurable electronics, a new problem is emerging. Data forms a vital part of these systems and is relied upon for their safety. Systems that are functioning "correctly" according to their design may become unsafe if given the wrong configuration or input data.

Concern about data-driven systems has recently been highlighted by articles in the Safety Critical Systems Club newsletter. Neil Storey argues that data is often ignored or overlooked at the initial stages of safety analysis and hence the familiar tools of system safety engineering are not applied to the data components of systems. He concludes that "there is no established 'best-practice' in this area and no meaningful guidance within the various standards" [46]. The sentiment is echoed by Felix Redmill, who bemoans the lack of available literature providing guidance on the preparation, validation, security, availability and integrity of data [42].

Storey's solution to the problem is for system designers to identify data as a system component, allowing consideration of the system-level effects of this component and subsequent application of hazard analysis and other safety engineering techniques. This idea gives the starting point that motivates this project: to improve hazard identification techniques so data is more likely to be identified as a potentially hazardous system component, and so that more of the hazards associated with data can be identified early in the design process.

3. Method

The aim of this project is to improve safety analysis so that accidents resulting from data issues may be avoided. To achieve this, the following method has been used:

- Literature review.
- Examination of the definition of data and discussion of related issues.
- Production of a taxonomy for data faults.
- Analysis of published accident and incident reports.
- Classification of data-related faults according to the taxonomy.

A literature review was carried out (section 4 of this report), to attempt to determine the current state of the art for safety analysis of data in systems, and determine where there is scope for improvement. This was divided into two areas: standards and published papers. It has also served to identify potentially useful techniques that can be built on by this work.

The literature review identified that while few standards are focussed entirely on the safety of data, guidance on safety assurance for systems that use data can be drawn from a variety of different sources. However, the review did identify that there is a gap in the guidance concerning hazard analysis for data.

Section 5 of the report looks at how that gap can be filled, by developing ideas about what makes up data, and what sort of problems can arise with it. In Section 6 this is developed into a taxonomy of data faults, drawing on previously published work and new material. The taxonomy is intended to be used as a checklist to guide hazard analysis.

In Section 7, a selection of accident and incident reports are presented, where the author has identified that safety-related occurrences have been caused or influenced by data considerations in the real world. Subsequent to the development of the data fault taxonomy, each accident was classified to show the type of data fault involved. This is important to help judge whether the proposed taxonomy is capable of being used in practice. As recommended by Holloway and Johnson [29], accident reports from official investigatory sources have been used as the basis from this work.

4. Literature Review

4.1. Standards

"The nice thing about standards is that you have so many to choose from."

Andrew S. Tanenbaum [50]

4.1.1. IEC 2382 (1993)

In ISO/IEC 2382, Information Technology Vocabulary, "*data*" is defined as "a reinterpretable representation of information in a formalized manner suitable for communication, interpretation or processing", where information is knowledge that has a contextual meaning. The definition of data is separate from that of "*software*", which consists of "programs, procedures, rules and associated documentation" [22]. While using similar definitions, later standards often treat data as part of software.

4.1.2. DO-178B (1992)

RTCA DO-178B: Software Considerations in Airborne Systems and Equipment Certification [2] uses the term data in a generic sense to mean information about or generated by the software development process. It does include data within its definition of software, but focuses only on the original airworthiness certification. Because it treats user-modifiable data as an operational aspect of software, it states that data certification is out of scope. However, the inclusion of this statement as specific example in the text hints that the authors believe that such data does require certification (albeit outside DO-178B). This can be explained by the fact that the DO-178B was designed to be used in conjunction with a suite of other documents such as ARP 4761 and DO-254, which would set requirements for other areas of the system development lifecycle [43].

The standard guides that from the planning stage software should be designed using fault tolerance techniques to tolerate errors in input data. Where user-supplied software or data ("*Field-Loadable Software*" (FLS)) can be added after installation, DO-178B gives guidance that the system should be able to detect corrupt, partially loaded or inappropriate data. Once such a fault is detected, the software should also be required to react appropriately. This should be checked using robustness test cases, which should include the possible failure modes of incoming data. The standard does not give guidance on how these failure modes should be determined.

DO-178B also highlights the potential for a new load of data to either change the configuration of the software (outside the configuration management process), or impair the ability of the system to determine its configuration. Either effect could impact the ability of the system to verify that it was operating in a certified configuration. There should also be protection to prevent the non-modifiable parts of the software being unintentionally changed by user-loaded data, or the modifiable parts being changed in a way that was not designed by the software supplier.

The lack of treatment of data assurance issues in DO-178B has not been noted as an issue in work published by the ASSC. It was not mentioned at all in a report looking at using DO-178B development as evidence for safety under UK military standards [43], and only in passing in guidance on applying the standard [25]. This guidance has a section on "additional considerations" for FLS. It says that FLS elements must meet the safety and security requirements of DO-178B, should be included in a suitable configuration management system and should be subject to a full Safety Assessment, with elevation to the Safety Working Group, if they need to be changed. It also suggests that FLS elements should be tested on the target computer, to show that the data is not corrupted and verify the algorithms. It is unclear whether this testing should be during preparation of the FLS, at runtime (it gives an example of using cyclic redundancy checks to detect corruption), or both. Verification of the algorithms embedded in the FLS might well require off-line testing on representative hardware, or use of static analysis techniques. The guidance also mentions that data integrity checks for executable code and data integrity tools should be described as part of a system's configuration control information, however this seems to

be more to do with ensuring that the software can be consistently reproduced than ensuring the safety of working data. While the guidance document does not add much guidance about data to that already contained in DO-178B, it asserts that FLS will generally only be used on systems of low safety criticality. It can be inferred from this that meeting the necessary safety requirements for field loadable software (including data), is seen as especially difficult or onerous.

4.1.3. DO-200A (1998) and DO-201A (2000)

As mentioned in section 4.1.2, DO-178B was written to form part of a wider suite of standards and guidance. Two other RTCA standards are relevant here: DO-200A, Standards for Processing Aeronautical Data [3] and DO-201A, Standards for Aeronautical Information [4]. The author has not had access to these commercial standards during the preparation of this report, however they are described by Faulkner and Storey in [21, 48].

According to Faulkner and Storey, DO-200A relates to air navigation data. It introduces integrity requirements for data and defines "qualities" or attributes, including accuracy, resolution, assurance level, traceability, timeliness, completeness and format [21]. DO-201A appears to describe the data requirements themselves, in terms of these qualities [48].

DO-200A mentions a need to establish the adequacy of relationships between data items, which Faulkner and Storey interpret as implying that the such relationships will require testing. The standard also provides a framework for describing the "data supply chain", a description of how data is generated, processed, passed through intermediaries, and eventually delivered to its end user. While the data chain may contain many links, the responsibility for satisfying the integrity requirements rests with the end user of the data [21].

4.1.4. Def Stan 00-55 (1997)

Def Stan 00-55 Issue 2, Requirements for Safety Related Software in Defence Equipment (now obsolescent), included data in its definition of software: "Computer programs, procedures and associated documentation and data pertaining to the operation of a computer system". However, it treated data separately in the text of the standard, which required that where the safety properties of such software relied on data, "the characteristics and properties of the data shall be formally specified and designed". Further clauses ensured that data was considered through the design process. They included requirements for traceability of the data design back through to the specification and requirements, use of strongly typed languages (to help prevent inappropriate low-level processing of data), inclusion of data use analysis in the static analysis of safety-related software, demonstration that the actual data to be used had the properties required for safety and use of representative data in validation tests [35].

The guidance given in Def Stan 00-55 focuses purely on how data is handled within a safety-related system. Beyond including "the procedures for and format of any parameters or data to be entered by the user" in the appropriate user manual [34], there is no guidance on how data should be prepared, or how the safety properties of data were verified.

4.1.5. IEC 61508 (1998)

While Def Stan 00-55 was a fairly prescriptive standard, IEC 61508, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems, is more goal-based[1]. It makes little distinction between data and software, which it defines as an "intellectual creation comprising the programs, procedures, data, rules and any associated documentation pertaining to the operation of a data processing system" [14]. "Data" and "data processing system" are not defined in the standard. The commonality between data and software is reinforced by a mandate that the requirements for software design and development "shall, in so far as it is appropriate, apply to data including any data generation languages" [13].

[1] IEC 61508 was produced in 1998-2000, but has subsequently been adopted as a European and British Standard. The BS EN version dated 2002 is cited in this report, which introduces minor corrections to the original text.

Despite this focus of IEC 61508 on data as a component of software, in Part 2 it considers the need for hardware to prevent data errors, by ensuring that data is stored, retrieved, communicated and processed without (undetected) errors [12]. Guidance on achieving this is given in Part 7, which summarises many suitable techniques for detecting such errors [15]. Part 2 also places a specific requirement on the calculation of the probability of undetected failures of a safety function due to data communication issues. In placing this requirement, it gives a list of types of data error that could be introduced by communication [12].

It is not clear that the majority of IEC 61508 software lifecycle, described in Part 3, is particularly applicable to data, although section 7.9.2.13 gives requirements for data verification. This includes verification of structures of data, application data and modifiable parameters and the hardware and software associated with any data interface [13]. No guidance is provided on how these requirements might be achieved, and there is no mention of the processes used for preparing data. The software lifecycle may be usable for data that is produced at the same time as the rest of the software, or only modified in line with the software modification process, but it does not appear to be very applicable to data that is modified more often than the software. The implication is that when data is changed, the software (including the data) will require re-verification. This may include static analysis or, during later lifecycle stages, testing [13]. Even with a single data configuration, it is unlikely to be feasible to test all possible combinations of inputs [9, 31].

4.1.6. SW01 (2003)

SW01, Regulatory Objectives for Software Safety Assurance in ATS Equipment, forms part of the Civil Aviation Authority's requirements for Air Traffic Services Safety [26]. It sets regulatory objectives for software safety. The standard is goal-based, requiring arguments and evidence to demonstrate that software is safe and referring out to other standards for guidance on how this might be achieved. It includes data in its definition of software, both that contained within programs and that on external media, if the data is necessary for the safe operation of the system. Despite this implicit acknowledgement that data issues may lead to unsafe operation, data does not appear to be included in the definition of a software fault, which is limited to defects in the program code. The only explicit requirement for data is in a sub-goal that requires that "the arguments and evidence, for the safety of the software in the system context, are from: a known executable version of the software and a known set of software products, data and descriptions that have been used in the production of that version". This implies that SW01 assumes data will be static for a given version of approved software. While this may be true for some slowly changing data such as geographical information, it is unlikely to hold for more frequently changing data like airline flight schedules. However the approach seems consistent with DO-178B (under which field loadable software, including data, should be certified).

4.1.7. Def Stan 00-56 (2007)

Issue 4 of Def Stan 00-56 superseded Def Stan 00-55, but took a radically different approach. Rather than being prescriptive, it is goal-setting. It gives generic requirements for safety to be demonstrated, for any type of system component (including data [36]).

The guidance part of the standard includes a section on the safety of systems containing "*Complex Electronic Elements*" (CEE). These elements include "databases, spreadsheets and other data" and "all forms of electronically executed algorithm(s) and associated data (such as configuration data, digital maps, lookup tables)". This guidance recommends that such elements should be subject to hazard analysis and have safety requirements, including integrity requirements, flowed down to them. There is then a discussion of the evidence that might be appropriate. In line with the goal-based philosophy of the standard, this is limited to types of evidence, rather than specific techniques [37].

4.1.8. Def Stan 00-56 Supporting Guidance (2008)

Subsequent to the withdrawal of Def Stan 00-55, the Ministry of Defence commissioned guidance [38] to assist contractors in producing safety justifications for Complex Electronic Elements (CEE) that would comply with the goals of Def Stan 00-56 Issue 4, as the guidance already contained in the standard was not perceived to be

sufficient. The pre-release version obtained by the author has a specific section on the Management of Assurance of Data and Communications. It categorises data as look-up tables, configuration data, communicated data and data that is input or output (including via human/machine interfaces). This data is considered part of the CEE, and as such should be subject to assurance processes to give confidence that it is adequately safe. The guidance suggests that for both the data itself and the lifecycle processes associated with the data, safety requirements should be set, hazard analysis should be performed, and the risks controlled.

The CEE Guidance also suggests specific types of safety requirement that might be appropriate for data, from which a list of classes of data error may be derived, as shown in section 6.3.3. It does not however propose any particular types of analysis technique as being specifically relevant to data.

4.1.9. Def (Aust) 5679 (2007)

The draft version of Issue 2 of Australian Defence Standard 5679, Safety Engineering for Defence Systems [10], explicitly recognises data safety issues. It includes "any system used to compute, compile, store or present data used to inform a safety-critical system" in its list of systems that are safety-critical and specifically recognises the contribution of "customisation data" to the safety case. Customisation data is defined as "data used to configure a system or component".

The draft standard requires -

- Design assurance activities during the installation process to ensure the correctness of customisation data;
- The architectural description to describe all uses of customisation data, and all tools used to produce it;
- Well-defined data format specifications and functional requirements for customisation data used by software; and
- A design verification that establishes that any instance of customisation data that satisfies these format and functional requirements will result in a safe component;

These requirements appear useful, although the final requirement may be hard to satisfy. The standard does not offer guidance on setting functional requirements for data and it may be hard to define safety requirements that are verifiable at design time. For instance, targeting data for a weapon system may be correctly formatted and represent a valid location, but be safe or unsafe depending on what occupies the targeted location at the time the weapon was armed. The standard does not seem to draw any distinction between configuration data that might be expected to be prepared once, then left largely unchanged, and more frequently changing operational data. The requirements seem to be worded more for the former type, but the inclusion of "weather conditions data" amongst the examples of customisation data hints that faster changing data is also within its scope.

It is interesting that there is no requirement for hazard analysis at the data level, hazards mainly being considered at the system level. The standard appears software-focussed and this philosophy presumably arises because software should not contain any known faults (assuming bugs are fixed on discovery). In contrast, data can often be expected to contain faults, e.g. due to interference on communication links.

4.1.10. GEIA-STD-0010 (2008)

GEIA-STD-0010 is a draft standard prepared by the G-48 System Safety Committee of the Information Technology Association of America's GEIA Group, an industry group representing companies servicing the US Government [16]. It is intended as a replacement for MIL-STD-882D. Version D of the MIL-STD had removed many of the prescriptive tasks found in previous versions, taking away the best practice element of the standard. MIL-STD-882D does not refer to data (in the context of this report) at all, although it does require hazard analysis of software [39].

Data is considered in GEIA-STD-0010 mainly as a type of asset – something that might be harmed by an accident. While this concept is useful in some respects, it will only drive safety analysis of the hazards to data if that data is already recognised as safety-critical. This may mean that data is omitted from the first pass of analysis. It also does not address situations where data is unharmed, but could still be hazardous. The standard does include "processing of safety critical data" as an area of software functionality that could contribute to a mishap, and says that this should be thoroughly analysed. However, this seems to be focussed on the processing of the data by software, not its correctness or content. Suggested areas for assurance of software integrity include verifying that data is used as specified and consistently, evaluation of data errors and protection of data from over-writing. No further guidance is given on how these might be achieved [16].

4.2. Published Papers

In 2000, Faulkner, Storey et al. commented on the safety management of data-driven safety related systems [18]. Their paper suggested that once data had been identified as an element of a safety-related system, it could be managed using "established safety management tools and techniques", but that guidance on such techniques is lacking in standards such as IEC 61508, Def Stan 00-55 and CENELEC prEN 50128. They identify that data errors can be both random (e.g. due to hardware failure) or systematic, and may be analysed either at the level of data types or individual data elements. They propose using a variation of Failure Modes and Effects Analysis for hazard identification and suggest range and consistency checks as examples of techniques which can manage certain classes of data error, once a potential hazard has been identified. The paper also suggests general requirements for the design of data structures and the processes used to produce, manipulate and test the data itself.

The focus throughout [18] is on configuration data. This is characterised in the paper as a static representation of the real world, as contrasted with dynamic data derived from human or sensor input. The paper seems to advise that the configuration data should be prepared at design time using suitable techniques to ensure safety, and then largely left unchanged. Although it recognises that there may be changes in the real world, which should be reflected in data, it does not suggest any method for such changes to be managed.

In 2001, Faulkner and Pierce [19] attempted to provide further guidance on suitable techniques for managing data once it has been identified as safety critical. Using the example of railway interlocking systems, the main concept of their paper is a division of configuration data into geographical elements and those elements which can be considered as programs written in a "limited variability language" defined by the system application. Traditional techniques can then be applied for the safety analysis of these programs. While the geographical data is regarded as static they note that modifications to the physical railway network must be reflected in the data used by multiple systems. Citing Harrison and Pierce [27], Faulkner and Pierce give nine classifications of error found in the data used to describe railway control systems and makes a distinction between detectable and plausible errors. While a system is able to deduce the presence of detectable errors, plausible errors can normally only be detected if the correct value of the data is already known.

To help increase the range of errors that are detectable, Faulkner and Storey introduce the concept of a data model in their 2001 paper [20]. Increased detection is achieved by structuring the model such that consistency checks can be made to identify errors in otherwise plausible data. The data model is designed with data validation in mind, and includes built-in rules to support this checking. Faulkner and Storey cite Welbourne and Bester's division of data into calibration, configuration and functionality data [53], and develop this into a generic data model including a description of the application instance, a command schedule, a description of the system's current status and a set of operating conditions. Faulkner and Storey [20] also identify a further set of five types of data error relating to data from external sources, in addition to those they previously cited from Harrison and Pierce. By considering these potential errors in the analysis of the data model, they intend that the model can be structured to allow avoidance of safety-related failures.

Tillotson [52] wrote in 2001 about safety-related management information systems in general, making some useful observations with respect to data. He points out that generation of data may often be divorced from its use, potentially with a large number of intervening systems. This has several implications. Those responsible

for generating the data may be unaware of the use to which it will be put or any associated risks. Hence they will not necessarily treat the data in a manner appropriate to its safety importance. Also, those using the data may be unaware of its original source or the processes through which it has passed. This will reduce the level of assurance they can gain about the correctness of the data, which should in turn reduce the level of trust that can be placed in the system.

Tillotson's response to this is to recommend that the data life cycle is explicitly analysed, with a responsible individual or organisation being identified for each stage. Hazards are identified for each stage (although Tillotson does not mention any specific identification techniques), and mitigations proposed.

Faulkner attempts to put the data model concept described in his earlier papers into practice in 2002 [17], giving a railway-based example using the four areas of his data model (application instance, command schedule, status data and operating conditions), the fault categories mentioned in [27] and [20] and a new set of fault types concerning status data (which shows some apparent overlaps with the other categories). Potential data faults in each area are identified, listed by fault type. However, there is no explanation of the hazard identification process and it is not clear whether the lists of fault types are used as guide words to drive the analysis, or merely used to classify the identified hazards after a brainstorm.

In the same paper, Faulkner also builds on Tillotson's work [52], defining a hierarchical model of the railway enterprise illustrating layers of abstraction from hardware elements up to the corporate level. Each layer has different requirements in terms of the data they use and the timeframe over which the data is expected to be valid. The safety impact of low-level changes in data may only be detectable when their effect at higher levels of the hierarchy is considered. The model also makes the point that different levels in the hierarchy may not know how the data was prepared at lower levels, or how it will be used at higher levels.

Throughout the various papers referred to above, Faulkner et al. identify that although standards such as IEC 61508 require data to be addressed through a data lifecycle, they contain a lack of guidance on how data should be structured and prepared to avoid errors. There is an acceptance that data should be prepared in a manner commensurate with the Safety Integrity Level of the system, but no guidance on how this should be achieved.

In 2002, Storey and Faulkner conducted an informal survey of engineers in safety-related industries to investigate how safety-related data-driven systems were being implemented in practice [47]. They found that hazard analysis was not generally being applied to data, resulting in a failure to set integrity requirements or carry out other actions to mitigate the associated risks. Configuration data was not being produced in a way that would meet the software lifecycle requirements of IEC 61508 or similar standards. In particular, in many instances the configuration data could not be validated separately from the rest of the system, but changes in the data did not result in a whole-system revalidation.

Storey and Faulkner proposed that configuration data should be explicitly highlighted in IEC 61508, by the inclusion of a data lifecycle that would include the specification, development, production and validation of data. Separate lifecycle models would be used to highlight the process requirements for the generic system and for individual configurations [47].

Despite Storey and Faulkner's survey findings, there is evidence that data is being considered as a potential hazard area in various different technology domains and that some system developers are taking safety precautions as a result.

In 2000, Hollow, McDermid and Nicholson had developed a method for attempting to address certification of reconfigurable Integrated Modular Avionics (IMA) systems [28]. In these systems, multiple avionic systems are controlled by software tasks running on a shared pool of multitasking processors, rather than separate processors for each device. An advantage of this type of architecture is that in the event of failures, tasks running on a failed device may be moved onto other processors with spare capacity. However, there are innumerable different reconfigurations possible, and it is likely that the certification process will assurance that each allowable configuration is safe. The allocation of tasks to processors is controlled by a "blueprint" – configuration data that defines which configurations are allowable.

Hollow et al.'s paper implicitly accepts that the configuration data has an important contribution to system-level safety. It does not describe how the safety of individual configurations are validated, but gives a process that allows equivalent configurations to be identified, such that when one safe configuration is known, safety assurances can be read across to other fallback configurations, reducing the workload involved in generating and validating the blueprint.

A workshop commissioned by the Ministry of Defence (MOD) in 2003 recognised that safety requirements would need to be developed for IMA systems. These would allow determination of rules for generation of blueprints. The workshop also questioned whether blueprints themselves would need validation, or whether it would be sufficient for them to be produced using validated tools [33].

In a different area of aviation, Simpson and Stoker recognise potential hazards caused by data when the note the challenges of terrain and obstacle data for navigation of Unmanned Air Vehicles [45]

The importance of data is also recognised in the railway domain by authors other than Faulkner and his associates. In 2003, Frazer et al. described the European Rail Traffic Management System (ERTMS) project [23], where a similar approach has been taken to that proposed by Faulkner et al. The hazards posed by data were identified from the inception of the project, leading to the use of an explicit data lifecycle and categorisation of types of data accorded to factors such as use and data lifetime. The consequences of failures of different types of error were identified, leading to an understanding that some failures could lead directly to an accident through a sequence of normal events, whereas others would also require an error on the part of the human operator of the system. Based on this distinction and the tolerability of the potential consequences, Safety Integrity Levels are allocated to different types of data. The paper then describes in generic terms the definition of a data supply chain, giving guidance on implementation, tools and techniques and interpretation of standards including IEC 61508, Def Stan 00-55 and RTCA DO-178B.

In 2006, Short examined the safety assurance of railway interlocking data [44]. His paper describes the data preparation process, identifying the points at which errors may be introduced and error checking can be performed. Like Faulkner, Short recognises that because data is viewed differently to software by developers, standards such as IEC 61508 or the railway-specific EN 50128 are not applied to its preparation. He suggests an approach to gaining assurance that the data is correct through a combination of four methods: testing, inspection, data flow analysis and formal proof.

In his 2007 masters report, Templeton analyses the requirements of Def Stan 00-55 and finds that its requirements for software can be applied analogously to the safety management of data. However, he identifies that there is a need for various complementary types of evidence, to give confidence in the argument for the safety of the data. A particular gap he identifies lies between formal software specifications and implementation of data structures. He proposes filling the gap with a new formal specification language for data structures which would give traceability to requirements and support rigorous analysis and testing [51].

4.3. Summary of Literature

Many of the published papers in the area of safety of data have been written by Faulkner, Storey, or both. The central theme of their work is that there is a lack of guidance in standards to support analysis of the contribution of data to safety. While most current standards for software or systems safety do not explicitly deal with data as a specific topic, there is a growing, if fragmented, body of work giving guidance on how to manage the safety impact of data.

DO-200A is a standard explicitly dedicated to data and appears to have some utility outside the aviation domain for which it was written. Storey and Faulkner have demonstrated that the lifecycle approach of IEC 61508 could be developed and extended to cover data explicitly [47], and a similar lifecycle approach has been used by Frazer et al. [23]. Templeton has shown that Def Stan 00-55 can be applied to the production of data and described detailed techniques for specifying data structures in a formal manner [51]. Assurance of data is now included as a specific area of concern in the new guidance supporting Def Stan 00-56 [38] and as specific requirements in Def (Aust) 5679 [10].

However, all of the literature reviewed considers data during its lifecycle: preparation, communication, processing and eventual use in a system. The focus is on ensuring that data structures and individual data elements are correct and unmodified. Despite exhortations that hazard analysis should be performed on data, none of the material reviewed by the author appears to offer any guidance on how this might be carried out. In those areas where data has been identified as possibly hazardous, this appears to have been "obvious". There appears to be a gap in the literature for a bottom-up method of identifying what could go wrong with data that would be potentially hazardous.

5. Discussion

5.1. Definition of Data

It can be hard to define precisely what is meant by data. The definition given by IEC 2382 is a good starting point: "a reinterpretable representation of information in a formalized manner suitable for communication, interpretation or processing". It is useful to consider the elements of the definition in turn.

"Reinterpretable" implies that data is designed to be used. It also implies that data is non-transient. It must persist for some time to allow its interpretation. "Representation" implies that data is not pure information in itself, but a construct that has been deliberately prepared. "Information" is defined by IEC 2382 as "knowledge concerning objects, such as facts, events, things, processes, or ideas, including concepts, that within a certain context has a particular meaning".

Taking these elements together, we can distinguish between data and real-world information. Real-world objects have attributes or properties, such as temperature, mass, position, etc., which may either be static or change over time. Once these values have been measured and the information taken into a system, the information becomes divorced from its real-world source, and can be considered data. This gives rise to the first generic failure mode of data: that its meaning may be incorrect with respect to the real object that it represents.

The next element in the definition of data is the "formalized manner suitable for communication, interpretation or processing". "Formalized" has the implication that data should have some particular format or syntax. It also implies a second generic failure mode: that the representation of the data, rather than its meaning, may be flawed and unsuitable for use.

While data is often thought of in terms of programmable electronic systems, it should be noted that nothing in the above definitions is specific to either electronic systems or software. The concept of data is applicable to systems in general that work with information, regardless of the technology used to implement them.

In the rest of this report, "*data element*" is used to refer to a representation of a single unit of information. A group of such elements present in a system at a given time is a "*data set*". Data elements may be transmitted as a "*data packet*" or in a continuous "*data stream*". Data packets and streams may contain partial or complete data elements or sets.

5.2. Taxonomy of Data Types

The author set out to derive a taxonomy for types of data but, after reviewing literature in the field, came to the conclusion that only broad classifications can usefully be made. Beyond this, data appears to have various properties or characteristics which vary independently, making a rigid taxonomy unhelpful.

Data can be divided into three broad categories, according to the type of information it holds: **objective** (describing a system's environment), **intent** (describing how a system will behave), and **third party** (of no direct relevance to a system).

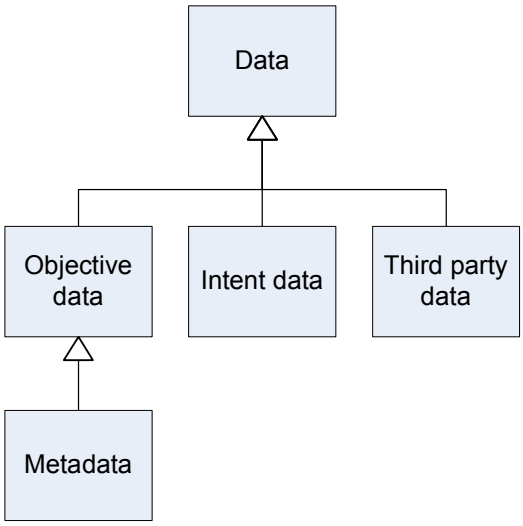

Figure 1. Decomposition of types of data.

The classification of a particular piece of data is dependent on viewpoint. A computer program may be seen as intent data by the system intended to run it, but objective information forming part of the environment of another system. Equally, either objective or intent data may be processed or communicated as third party data through various systems before reaching its end user.

5.2.1. Objective

Objective data carries information about objects, both internal and external to the system. The word "object" is used to refer to not just physical objects but, as per IEC 2382, facts, events, things, processes, ideas and concepts. Objective data describes the system and the world it inhabits. Examples of could include a railway timetable, an address book, a ship loading model, or a navigational chart. Configuration data that *describes* how a system is configured (e.g. what physical hardware is present, how nodes are connected together) is objective.

Objective data may be hazardous if it does not accurately represent the information in question, because systems will make decisions that do not reflect the real state of the world. Data may be incorrect through flawed initial preparation. It can also become incorrect over time, either due to modification (intended or otherwise) or due to the real-world object changing.

An important sub-category of objective data is **metadata**. Metadata is data that represents information about data itself. Metadata can be used to describe the type of information that a data value represents, its format, where and when it originated and what its qualities are. Since metadata describes real-world objects (other data elements), it can be considered objective data. However, the objects that it describes may be objective, intent or third-party data.

5.2.2. Intent

Intent data tells a system how to behave. It does not correspond to any object outside the system, so cannot be directly verified. It might consist of parameters, instructions, rules or algorithms and might affect either the functional or non-functional behaviour of the system (what it is supposed to do, and how). Intent data ranges from very simple boolean configuration options, to complex programs. Using this definition, most software could be considered intent data. Configuration data that *determines* how a system should be configured, such as an IMA blueprint, is intent data.

There is some cross-over between objective and intent data. For instance, a definition of a communications protocol could be considered both objective data about how external objects will react and intent data about how a system should behave to interface with those objects.

5.2.3. Third Party

Third party data refers to data that is not used by a system, but is passed through as a payload. Although this data has no direct safety impact on the local system, there may be safety implications for client systems that use the data. For this reason, system designers need to know the safety requirements for third party data, so that they can ensure that a sufficient level of integrity is preserved. An example of third party data would be the messages passed through a communication system.

5.3. Data versus Software

While IEC 2382 and the discussion in section 5.2 has considered software as a sub-classification of data, many of the other standards included in the literature review have included data in part of their definition of software. The situation is confused by the existence of programming languages such as Lisp, where the same structures are used to represent both program code and data, and data transfer formats such as JSON, which uses an executable subset of the Javascript language to represent data values [1]. It is worth examining this relationship in more detail.

Although software can be considered a representation of information (and hence data), it also forms part of the definition of a system. In standards such as DO-178B and Def Stan 00-56, which require configuration management, changes to software are counted as changes to the system, and as such would require safety assessment. When considered in this regard, intent data is considered "software" if it changes the basic function of a system, or "data" if it does not.

The distinction provided above is fairly arbitrary. In reality there seems to be a continuum between what might be considered "pure software" and "pure data", depending on the level of integration between the data and the information. Examples of different levels of integration are given in Table 1.

Level of integration	Description
Offline	Human interaction required for any use of the data by a system. E.g. hard copy charts or look-up tables.
Loadable	Data is stored externally to the system, but is available via a removable storage medium or communication link. The data must be presented to the system for use, e.g. by inserting a card or disk.
Configurable	Data is stored within the system, but is logically separate, and may be modified. E.g. software configuration files.
Pre-configured	Data is logically separate from procedure at design time, but is fully incorporated in the system and cannot be separated from it. E.g. header files defining constants that are compiled into software.
Implicit	Data is implicit in the definition of procedures. E.g. string literals or constant values hard coded into algorithms.

Table 1. Levels of integration of data into a system.

At the loosest level of integration, the system cannot directly use the data, and must request it through an intermediary. While the implications of how the data is used may not be obvious, such data is obvious as an input to the system.

At the tightest level of integration, the direct effect of the data on the system will be more obvious, but it may not be apparent that the data is in fact data, rather than simply a part of the system design. This may mean that the link between the data and the real world information it represents is not recognised and preserved.

The level of integration of the data has implications for the assurance of its integrity. The tighter the data is integrated with the design, the more likely it is to be included in the formal system development programme, and hence exposed to safety assessment.

Level of integration does not however correlate with the level of influence of data on the behaviour of a system, and hence its potential to be hazardous. Software loaded via external media or communication links (e.g. viruses) can still affect the functionality of a system.

5.4. Sensor Input, Communications and Rate of Change of Data

Many systems gather information from sensors and it could be asked whether there is a difference between this type of information and the information provided by data. If the information is used directly by the system, and not stored or communicated, then it does not form data. If the information is represented in a reinterpretable, formalised form, then it would fall within the definition of data. In some cases there may be little benefit in giving such data special consideration, as sensors form part of the basic functional design of a system, and engineering in general has long focussed on constructing systems that respond correctly to their inputs.

However, when sensor data is communicated to other systems or stored internally for later use, it is within the scope of this report. An example is in learning systems, which build up a database of information as they operate. These systems may suffer from latency, when the data they have gathered becomes out of step with the real world. Anecdotally, some cars with early computerised fuel injection systems only measured air pressure at start-up. They would stall if driven to a significantly higher altitude, as the recorded value no longer gave the correct fuel/air mix for the local air pressure.

This example raises the question of rate of change of data. If real-world information changes slowly, then data may remain valid for a long period of time without update. In this case, it may be practical to put a large degree of effort into assuring that the data is correct, and that the system will function safely with those data values. It also becomes practical to integrate the production of the data tightly into the system development process.

If the information changes rapidly, then data must be updated frequently in order to remain valid. This is likely to mean that it is not practical to integrate production of the data into the system development, as it becomes impractical to revalidate the system for each new iteration of data. It may also reduce the range of measures that are cost-effective for assuring that the data is correct. This is particularly relevant in the case of sensor data, or data received over communication links. This may change frequently, and will not be available at design time.

As well as the frequency of changes to information, it is important to consider the magnitude of the variation. Some types of information vary continuously, such as those representing physical quantities (e.g. temperature). If such information is varying slowly, then data about it may continue to be valid for some time before becoming unacceptably inaccurate. Other types of information are discontinuous and, although they may change infrequently, the slightest change may render the data totally invalid. E.g. a single-bit change in the address of a network node would make that node unreachable.

With continuously varying data, it is therefore necessary to consider how much the underlying information would be allowed to change before the data became unacceptable, and hence the necessary update rate for the data. With discontinuous information, more thought must be given to how to ensure data reflects important changes. Changing the sampling rate may not be the most efficient way to achieve this, but a potential approach is to consider how long the system can tolerate using incorrect data.

It is also possible that the rate of change of information may vary. In a military context, data that might otherwise be expected to change very slowly, such as the location of political boundaries or infrastructure like bridges, may change very quickly due to battle damage. Similar effects might occur due to natural disasters, such as the Indian Ocean tsunami of 2004. In other contexts, a change of the operational environment of a system may require data to be sampled at a different rate.

5.5. Preparation of Data

During initial system development, configuration and test data will generally be produced by or with the assistance of the developer. At this stage, the developer has the opportunity to fine tune the data to ensure that it is correct and that it will work safely with the rest of the system. The developer also has great familiarity with

the intended use of the data and the way that this use is implemented by the system, potentially allowing data faults to be unconsciously avoided. However, once the system is deployed, the developer often has little further involvement with subsequent production of configuration data [47]. This raises the question of how sensitive the system is to subtleties in the data it uses. There is the potential that user-developed data which may superficially appear valid may cause unexpected problems at the system level.

Once a system is deployed, how data is prepared can vary greatly. In some industries where there is tight regulation or strong vertical integration between suppliers and end users, the system designer may be able to have a strong influence on the data preparation process. In such situations it may be possible for data preparation to remain part of the formal software lifecycle process.

In many areas it is more likely that there will not be strong organisational links between the designer and end user. However, designers can use a variety of techniques to retain control of the data:

- The data could be preconfigured, tested and validated for a particular application instance, which is then not user-modifiable. This could be used to customise a system for a particular customer on commissioning.
- User input could be restricted to a limited set of data values, e.g. by physically constraining the input mechanism. If the set of values is small enough, it may be possible to test that all possible combinations are safe.
- More freedom can be given to the user by defining constraints on safety, which are implemented in the system, allowing the data to be checked for safety at runtime.

In many situations, the designer has very little control over the preparation of input data so must assume errors will occur. Factors such as transmission errors or hostile action (not just a military concern in these days of terrorism and crime), may mean that faults in data can be expected on a regular basis, and systems must be designed to safely tolerate them. System designers must therefore aim to ensure that all potential hazardous data faults are detectable and recoverable. To guide this work, a useful tool would be a checklist or taxonomy of data fault types, to steer the hazard identification process.

5.6. Data and Risk Analysis

Traditional risk analysis is based on probability and a model that assumes that unwanted events are caused by failures or conditions that occur with a known random distribution. This model does not appear to be necessarily valid for systems whose behaviour is determined by data.

Some data processes will generate random errors, such as measurements of physical quantities, or transmission of a data stream over a lossy communications link, but other faults in data may be systematic, or unpredictable but without a statistically random distribution. Mechanisms exist for these problems to cause unsafe conditions, in the form of subtle data value faults that cannot be detected due to their plausibility.

Similar issues exist with software, where it can be argued that all faults are systematic, and the randomness of the outcome is only due to the randomness of the input parameters. However, it is normally possible to at least partially verify software before run-time.

Real data may not be available until run-time, which may mean that only limited testing and verification can be performed. Systems can perform run-time checks for consistency or formatting, but in many cases may need to rely on trust that the values they have been provided with will be safe.

6. Taxonomy of Data Faults

6.1. Generic Data Faults

The definition of data given in section 5.1 lead to several generic failure modes for data:

- Meaning incorrect with respect to real-world data. This can be caused by flawed preparation, or either the data or the real-world value changing after preparation of the data.
- Format incorrect. This can be caused by either flawed preparation, the data being changed (corrupted) after preparation, or the required format of the data changing without update of the data.

These failure modes are too generic to be of much practical use, but serve as high level categorisations.

6.2. Existing Fault Taxonomies

Many of the standards and papers included in the literature review contain descriptions of different types of data fault or issue. These are listed in this section for comparison.

6.2.1. Beizer's Bug Taxonomy

In 1990, Beizer published a hierarchic taxonomy of software bug types along with statistics on the proportion of each type in a sample of over 16000 bugs recorded prior to 1989. The taxonomy was updated by Vinter in 2001, who added new statistics covering nearly 1000 further bugs recorded between 1992 and 1998 [5].

Bugs relating to data were included as one of Beizer's nine top-level categorisations. The category was sub-divided into two areas: Data Definition, Structure & Declaration; and Data Access & Handling. Many of the lower-level items in the taxonomy are more relevant to how data processing is implemented in software, but a number relate to bugs in the data itself and hence are of interest here. These are shown in Table 2.

Category	Description
4212	**Wrong type:** object type is incorrect for required processing: e.g., multiplying two strings.
4214	**Type transformation:** object undergoes incorrect type transformation: e.g., integer to floating, pointer to integer, specified type transformation is not allowed, required type transformation not done. Note, type transformation bugs can exist in any language, whether or not it is strongly typed, whether or not there are user-defined types.
4216	**Scaling, units:** scaling or units (semantic) associated with object is incorrect, incorrectly transformed, or not transformed.
4234	**Constant value:** incorrect constant value for an object: e.g., a constant in an expression.
4262	**No such resource:** referenced resource does not exist.
4264	**Wrong resource type:** wrong resource type referenced.
4281	**Wrong object accessed:** incorrect object accessed: e.g., "X := ABC33" instead of "X := ABD33."
4282	**Access rights violation:** access rights are controlled by attributes associated with the caller and the object. For example, some callers can only read the object, others can read and modify, etc. Violations of object access rights are included in this category whether or not a formal access rights mechanism exists: that is, access rights could be specified by programming conventions rather than by software.
4283	**Data-flow anomaly:** data-flow anomalies involve the sequence of accesses to an object: e.g., reading or initializing an object before it has been created, or creating and then not using.

4284	**Interlock bug:** where objects are in simultaneous use by more than one caller, interlocks and synchronization mechanisms may be used to assure that all data are current and changed by only one caller at a time. These are not bugs in the interlock or synchronization mechanism but in the use of that mechanism.
4288	**Object boundary or structure:** access to object is partly correct, but the object structure and its boundaries are handled incorrectly: e.g., fetching 8 characters of a string instead of 7, mishandling word boundaries, getting too much or too little of an object.

Table 2. Selected categories from Beizer's Bug Taxonomy [5].

Data bugs in general made up 22% of Beizer's statistics and 8% of Vinter's. The proportions of bugs allocated to the "interesting" categories listed above were as shown in Table 3. Note that categories in which no bugs are found are not shown in the table.

Cat.	Description	No. bugs found		% bugs found		
		Beizer	Vinter	Beizer	Vinter	Total
4212	Wrong type	10	1	0.1%	0.1%	0.1%
4214	Type transformation	84		0.5%	0.0%	0.5%
4216	Scaling, units	237	4	1.5%	0.4%	1.4%
4219	Other type bugs	28		0.2%	0.0%	0.2%
4281	Wrong object accessed	244	22	1.5%	2.2%	1.5%
4282	Access rights violation	8		0.0%	0.0%	0.0%
4283	Data-flow anomaly	115	10	0.7%	1.0%	0.7%
4288	Object boundary or structure	115		0.7%	0.0%	0.7%
	Total:	841	37	5.2%	3.8%	5.1%

Table 3. Occurrence of data bugs in software.

Beizer and Vintner's data does not cover all the types of data fault covered in this section. It also considers all bugs, rather than just those with safety-related outcomes. However, the fact that approximately 1 in 20 software bugs were attributed to data issues means that this is a significant area, worthy of research.

6.2.2. Railway Industry Data

Faulkner has assembled various classifications of data errors from the railway domain [17], which are reproduced below in Table 4 to Table 6.

Omission	An infrastructure entity is not present in the control system data set.
Spurious data	A non-existent entity is present in the data set, this may also include duplicated entities.
Positioning errors	For example an entity is represented and addressed correctly, but its physical position is incorrect.
Topological errors	All entities are present, but they are connected in a way which may be plausible, but incorrect.
Addressing errors	An entity is correctly located and labelled but is connected to the wrong field devices.
Type errors	An entity is connected and labelled correctly but is recorded with an incorrect type identifier.
Labelling errors	An entity is located and addressed correctly, but is assigned the wrong label in the data model.
Value errors	A scalar attribute of some entity in the configuration data has the wrong value.

Table 4. Data faults in railway infrastructure data (Harrison & Pierce) [27]

Existence	A data reference provided by one external information system cannot be fulfilled by another information system.
Reference error	The wrong data reference is provided resolving information which does not represent the required train.
Availability	An external information system is not available (off-line) at the time the information is requested.
Inconsistent	Data requested from more than one external information systems is inconsistent.
Timely	Data is not supplied until after the event.

Table 5. Data errors from external errors (Faulkner & Storey) [20]

Mode fault	The operational mode for one or more components is incompatible with the current operational mode.
Sequential fault	Data presented to the railway control system is "out of sequence".
Combinational fault	Data presented to the railway control system is incomplete to meet the requirements of a predetermined condition.
Propagation fault	Data transmitted to the railway control system is changed or corrupted on receipt.
Timing fault	Data is presented to the railway control system earlier or later than expected by the system.
Volume fault	The railway control system is flooded with or starved of data.

Table 6. Data faults in status data (Faulkner) [17].

It can be seen that there are some overlaps between these classifications. For instance, "omission", "existence" and "availability" are functionally equivalent. Although they have different causes, in each case the required data is missing.

6.2.3. SHARD Guidewords (Pumfrey 2000)

In his doctoral thesis [41], Pumfrey cites classifications of computer systems failures due to Ezhilchelvan and Shrivastava, and Bondavalli and Simoncini, then develops a set of guidewords for hazard analysis of software. These guidewords may have wider application to data.

Ezhilchelvan and Shrivastava's classification is a hierarchy, based on a basic "byzantine" failure, which could include any type of failure mode. A specialisation of this is "emission", where the output has the wrong timing, value, or both. Value and timing faults are the next level of the hierarchy, and the final case is "omission", where no service is delivered.

Bondavalli and Simoncini divide failures in to the timing and value failures. Timing can be early, correct, late, or infinitely late. Values may be correct, subtly incorrect, coarsely incorrect, or omitted.

A key feature of both classifications is the distinction between the time and value domains. Bondavalli and Simoncini also make a useful distinction between failures that can and cannot be detected by the recipient. Although these classifications were designed for computer systems in general, they still appear to be useful for application to data.

Pumfrey went on to develop guidewords, partly based on these classifications, for the Software Hazard Analysis and Resolution in Design (SHARD) technique, shown in Table 7. This list has the benefit of being very simple, while addressing three problem domains: provision, value and timing.

Omission	The service is never delivered, i.e. there is no communication.
Commission	A service is delivered when not required, i.e. there is an unexpected communication.
Early	The service (communication) occurs earlier than intended. This may be absolute (i.e. early compared to a real-time deadline) or relative (early with respect to other events or communications in the system).
Late	The service (communication) occurs later than intended. As with early, this may be absolute or relative.
Value	The information (data) delivered has the wrong value.

Table 7. SHARD Guidewords [41].

6.3. Fault Taxonomies extracted from Standards

Many standards either give a description of desirable properties of systems, or particular hazards that should be guarded against. By extracting these, further (partial) taxonomies may be derived.

6.3.1. IEC 61508

Part 2, Section 7.4.8 of IEC 61508 requires a system designer to take into account the following types of error in data communications [12]:

- Transmission errors
- Repetitions
- Deletion
- Insertion
- Resequencing
- Corruption
- Delay
- Masquerade

6.3.2. DO-200A

According to Faulkner, DO-200A provides seven "qualities" of data: Accuracy, Resolution, Assurance Level (confidence that the data is not corrupted while stored or in transit), Traceability, Timeliness (level of confidence that the data is applicable to its intended period of use), Completeness and Format [21].

Each of these qualities could be the basis for setting a safety requirement. By negating each quality, a list of associated faults may be generated that could lead to hazardous consequences:

- Inaccurate data value
- Insufficient data resolution
- Data altered after preparation
- Unknown provenance
- Untimely data

- Incomplete data
- Incorrect format

"Data altered after preparation" would mean that the true provenance of the data was not known, and could result in any of the other types of fault. Although "unknown provenance" of data does not immediately imply a hazardous condition, failure to be able to trace the data back to its source may lead to decisions being based on inappropriate data, such as default values, training or test data that had been allowed to leak onto a live system.

6.3.3. MOD CEE Guidance

In the Ministry of Defence's guidance on Safety Assurance of Complex Electronic Elements, Section 2 gives example safety requirements for data. These include valid data types, ranges, relationships, and timeliness [38]. As with the DO-200A qualities, these may be negated to give the following fault types:

- Type errors
- Range errors
- Incorrect relationships between successive data items (e.g. sequence numbers skipped or out of order)
- Untimely data
- Invalid relationships between different data items (such as whether a date of 31 is valid in a particular month);
- Invalid relationships between data items and the state or internal data of the CEE (for example in a database application, the data item must exist in the database before a request to delete it can be processed).

In addition, the guidance specifically highlights a further set of failure modes specific to data communications:

- Corruption (random and systematic)
- Repeated receipt of data
- Loss of data (random, systematic, and permanent)
- Data delayed (variable or consistent delays) or out of sequence
- Deliberate attack (e.g. denial of service, viruses or jamming)

6.4. A Combined Data Fault Taxonomy

The fault taxonomies presented in sections 6.3 and 6.4 provide 61 candidate fault types. Within these types there are a number of duplications and several areas where similar category names have different meanings. They have been combined into a new taxonomy for use as a checklist for hazard identification, shown in Table 8.

A hierarchic organisation was used to give a balance between brevity, for simplicity and ease of understanding, and comprehensiveness, for effectiveness as a checklist. For this reason, four basic categories of fault have been chosen, and the hierarchy developed to identify specialised forms of faults that might not otherwise have been apparent. The indentation in Table 8 indicates levels of specialisation.

Serial	Fault Category	Description
1.0.0	Meaning	Meaning of data incorrect
1.1.0	Value	Value of data incorrect
1.1.1	Meaningless	Value not capable of interpretation
1.1.2	Inaccurate	Value valid but wrong
1.1.3	Association error	Value correct, but referring to wrong object
1.2.0	Insufficient resolution	Data does not reveal artefacts of interest
1.2.1	Aliasing	Data implies non-existent artefacts
1.3.0	Ambiguity	Data not capable of consistent evaluation.
1.3.1	Multiple interpretation	Data can be understood to mean different things
1.3.2	Violation of uniqueness	Data refers to multiple objects, instead of one.
1.3.3	Lack of precision	Uncertainty in measurement of data
1.4.0	Inconsistency	Disagreement between data items
1.4.1	Between instances of same data	e.g. different sources of same information differ
1.4.2	Between successive data items	e.g. sequence or pattern not followed
1.4.3	Between different data items	e.g. mutually exclusive configuration options
1.5.0	Omission	Element left out of data set
1.5.1	Incomplete data	Required data partially missing
1.6.0	Commission	Additional elements in data set
1.6.1	Repetition	Element inadvertently repeated in data set.
1.6.2	Data overrun	Unrequired partial data additional present
2.0.0	Format	Data is not formatted correctly
2.1.0	Wrong scaling or units	Data uses wrong measurement units
2.2.0	Wrong datum	Measurement not made from correct baseline
2.3.0	Wrong type	Data uses incorrect physical representation
2.4.0	Data out of range	Data value out of range for data type
2.5.0	Wrong language	Wrong human or software language used
2.6.0	Wrong grammar	Data violates grammar or syntax rules of format
2.7.0	Data out of sequence	Data elements incorrectly ordered
3.0.0	Timing	Data does not arrive at the correct time
3.1.0	Omission	Expected data does not arrive from source
3.1.1	Availability	Data source cannot be contacted
3.1.2	Existence	Data source does not have data
3.1.3	Access	Data source will not grant access to data.
3.1.4	Loss	Data is lost en-route from source
3.2.0	Commission	Unexpected data arrives
3.2.1	Repeated receipt of data	Expected data arrives more than once
3.3.0	Early	Data arrives earlier than expected by system
3.4.0	Late	Data arrives later than expected by system
3.4.1	Validity exceeded	Data arrives later than expected by preparer
3.5.0	Data out of sequence	Data elements arrive in wrong order
4.0.0	Provenance	Data not from desired source
4.1.0	Masquerade	Data not from identified source
4.2.0	Unknown provenance	Source of data cannot be identified

Table 8. A new taxonomy for data faults.

The original 61 fault categories have been refined and new categories added, resulting in 43 new categories organised in four broad areas. The “meaning” and “format” faults identified from the definition of data have

been included as top-level faults. To these have been added timing, which is present in most of the other taxonomies studied, and provenance.

Most of Beizer's bug categories have been mapped into the new taxonomy, albeit often in more generic categories. However, "incorrect scaling, units" and "access rights violation" were unique to Beizer's taxonomy. "Type transformation" and "interlock bug" were omitted, as they appear to be more relevant to the processing of data than the data itself and the fault conditions caused by these processing failures if into other categories, such as "incorrect type" and "incorrect value".

Harrison & Pierce's railway infrastructure faults appeared quite domain specific, however they have all mapped into the new taxonomy. It is notable that Harrison's "Type" fault maps to a "Value" fault in the new taxonomy, because it is the value of a reference to a real-world object that is wrong, rather than the data having the wrong type itself. "Positioning" faults also map to "Value" faults, although these are interesting because the cause of the fault is the real world being implemented the wrong way, rather than the data. However, the hazard is the same in either case.

All Faulkner and Storey's external data faults map into the new taxonomy. This set of faults was useful in prompting thought about the accessibility of data. It raises questions about whether data exists, whether a source will provide it and whether the necessary communication links work.

Most of Faulkner's status data faults map into the new taxonomy, with the exception of "*propagation fault*". This category is too generic, and could be a cause of many different types of fault. "*Volume faults*" map to a special case of "*early*" or "*late*" data but flooding has not been specifically included. In this case the fault lies with the inability of the system to process the data, rather than a flaw in the data itself.

Pumfrey's SHARD guidewords all map into the new fault taxonomy at a fairly high level. "*Early*" and "*late*" have been grouped together under a "*timing*" category. "*Omission*" and "*commission*" have been duplicated, as they have different interpretations in the meaning and timing domains. In the meaning domain, it is implied that a data set exists at the correct time for use, but either expected values are not present, or there is additional unexpected information. In the time domain, omission and commission are interpreted as meaning that whole messages or sets of data are either missing or unexpectedly present.

Most of the communications factors listed in IEC 61508 map into the new data fault taxonomy, apart from "*corruption*" and "*transmission errors*", which are too generic to describe the actual problem with the data. An interesting contribution from this standard is "*masquerade*", a violation of traceability where data from one source appears to be from another, either accidentally or through malicious intent.

DO-200A has been a useful input into the new data fault taxonomy, for introducing a number of relevant areas that are not directly related to the accuracy of a data value. Faults derived from all the DO-200A qualities have been mapped into the taxonomy, with the exception of "*data altered after preparation*" (derived from assurance level, which gives the level of confidence that data has not been modified in storage or transit). This was seen as a potential cause of a range of different faults in the "*meaning*" or "*format*" categories.

The faults derived from the MOD's guidance for complex electronic elements are all relevant to data and have mainly been mapped into the new taxonomy. The guidance was particularly useful in pointing out the various types of fault relating to associations between data items. These inconsistencies can lead to the detection of fault conditions that would otherwise appear as plausible data values. "*Corruption*" and "*deliberate attack*" have not been included, as they are potential causes of many different faults, rather than faults in themselves.

The derivation of each fault category is given in Table 9. Appendix A shows in detail the coverage of the new categories over those given in the source material.

Serial	Fault Category	Derivation
1.0.0	Meaning	New category, arising from the generic definition of data and a need for a broader category than just “Value”.
1.1.0	Value	Faults concerning the value of the recorded data are found in most sources.
1.1.1	Meaningless	New category, included to highlight a special case of corruption where the data cannot be interpreted as any valid value.
1.1.2	Inaccurate	From DO-200A. Cases where the value is plausible, but wrong.
1.1.3	Association error	Included to capture addressing and labelling errors (Harrison & Pierce), reference errors (Faulkner & Storey), and wrong resource type or object accessed (Beizer). These are faults where the data refers to the wrong object, but is not necessarily incorrect in reference to the object it was based on.
1.2.0	Insufficient resolution	From DO-200A, used here to focus on resolution across the sample space (e.g. time, distance, count between samples), rather than in the quantity being measured.
1.2.1	Aliasing	New category, introduced to capture resolution issues that create artefacts, rather than hide them.
1.3.0	Ambiguity	New category, introduced to capture the idea that use of data should be deterministic.
1.3.1	Multiple interpretation	New category, where data could have different meanings.
1.3.2	Violation of uniqueness	New category, where a one-to-one relationship unintentionally becomes one-to-many.
1.3.3	Lack of precision	Similar to resolution, but focussed on the measurement domain, rather than sample space.
1.4.0	Inconsistency -	Introduced to group relationship-focussed faults.
1.4.1	- between instances of same data	From Faulkner & Storey.
1.4.2	- between successive data items	From the MOD CEE Guidance.
1.4.3	- between different data items	From the MOD CEE Guidance, including Harrison & Pierce’s topological errors and Faulkner’s mode faults. Deals both with internal and external data sources.
1.5.0	Omission	From Pumfrey, Harrison & Pierce and IEC 61508 (deletion).
1.5.1	Incomplete data	From DO-200A, Faulkner and Beizer.
1.6.0	Commission	From Pumfrey, Harrison & Pierce (spurii) and IEC 61508 (insertion).
1.6.1	Repetition	From Harrison & Pierce (spurii) and IEC 61508.
1.6.2	Data overrun	From Beizer’s “object boundary or structure”. In contrast to incomplete data, surplus partial data.
2.0.0	Format	From DO-200A, and the basic definition of data.
2.1.0	Wrong scaling or units	From Beizer.
2.2.0	Wrong datum	New category, introduced to capture problems where measurements use different baselines.
2.3.0	Wrong type	From Beizer and the MOD CEE Guidance.
2.4.0	Data out of range	From Beizer and the MOD CEE Guidance.
2.5.0	Wrong language	Introduced to capture differences between dialects, both human and electronic.
2.6.0	Wrong grammar	Introduced to capture faults where the data structure is not properly constructed.
2.7.0	Data out of sequence	Implied by various sources, although most concentrate on sequence in the time domain, rather than the order of data in a static

		structure.
3.0.0	Timing	Implied by most sources.
3.1.0	Omission	From Pumfrey.
3.1.1	Availability	From Faulkner & Storey (external data source unavailable).
3.1.2	Existence	From Faulkner & Storey (reference cannot be fulfilled).
3.1.3	Access	From Beizer.
3.1.4	Loss	From the MOD CEE Guidance.
3.2.0	Commission	From Pumfrey.
3.2.1	Repeated receipt of data	From the MOD CEE Guidance and IEC 61508.
3.3.0	Early	From Pumfrey.
3.4.0	Late	From Pumfrey, Faulkner & Storey, IEC 61508 and the MOD CEE Guidance.
3.4.1	Validity exceeded	From DO-200A (timeliness).
3.5.0	Data out of sequence	From Faulkner, IEC 61508 and the MOD CEE Guidance. Also implied by Pumfrey (who includes relative timings in early / late) and Beizer, for data flow anomalies.
4.0.0	Provenance	From DO-200A (unsatisfactory traceability).
4.1.0	Masquerade	From IEC 61508.
4.2.0	Unknown provenance	From DO-200A (untraceable).

Table 9. Derivation of fault categories.

6.5. Discussion of Data Faults

Correctness is dependent on point of view. Data that is "correct" when prepared may not be correct when used, e.g. a quantity measured accurately in metres may be "incorrect" if the physical distance has changed before the data is used, or if the user was expecting a value in feet, or a measurement of a different object. Correctness is therefore relative, and is described in the notes below with respect to the intent of the consumer of the data, at the time of consumption.

It should also be noted that the faults can be present at different levels of structure within data. Some faults, such as value faults, may be applicable to a single element of data. Others, such as inconsistencies or data out-of-sequence, only have meaning when multiple data elements are considered. A fault may also fall into more than one category, e.g. a wrong value in one element may also be classified as an inconsistency or format fault, if other data elements are taken into account. It should be an aim of system design that any viable fault in the data should be capable of being classified in at least one category of detectable fault.

6.5.1. Meaning

The most obvious and important type of fault in data is to do with *meaning*, the first category in the taxonomy. The meaning of data is at fault if it is incorrect with respect to the real world, if it does not correctly reflect the state of the objects in the real world (objective data), or the user's current intent (intent data). These faults may come about in three main ways. The data may be prepared wrongly from the start, it may be changed in a way that makes it wrong, or the real world may change, so that it no longer correctly reflects its subject or intent.

The key component of meaning is the *value* of the content of a data element. Value faults may be obvious, but can be subtle, when the presented value is wrong but plausible and hence hard to detect. Much of the emphasis in safety management of data is to try to construct data structures such that as many potential faults are detectable as possible. Often this is achieved by using metadata, formatting, or additional knowledge about the system, so that value faults cause inconsistencies or other types of detectable fault.

Some value faults result in a value has become *meaningless* (e.g. an undefined character) and are easily detected. Those faults where an incorrect value has a meaning can be divided into two further types: those where the value

is *inaccurate*, and those where the value is accurate, but does not refer to the object that the user of the data intended (*association* faults).

Various methods are available to protect against value faults caused by data changing. Often these involve adding in redundant information so that inconsistencies can be detected (e.g. duplicate records, or coding schemes where n-bit errors result in invalid or uniquely identifiable symbols), or metadata (such as checksums and hashing codes), to check whether some measurable property of the data has been changed before receipt. In the case of metadata, precautions are strengthened when the metadata can be transmitted separately to the data itself, removing a potential common cause of failure.

If data is prepared wrongly, there may be little that can be done to detect the fault, unless the same information is available from an independent source to allow consistency checks. However, formatting can be used to protect against some kinds of preparation error, and traceability or provenance can be used to give assurance that the data came from a high quality source.

The hardest value errors to check are those where the real world has changed, leaving the data stale. In these cases the value remains plausible, but has become incorrect with respect to the user's intent for what it was supposed to represent. These subtle faults can be protected against to a limited extent by changing the rate at which the data is sampled (which has resource implications), or adding validity metadata to the data to reflect the expected rate of change of the underlying information.

The *meaning* category of data faults was introduced as a high-level category because there are various circumstances where a data element may not be "wrong", but still does not convey the correct meaning. These include issues of resolution, interpretation and consistency.

The source information may be sampled with *insufficient resolution* to identify features that are important to a user. Consider the example of a digital imaging system used for object detection. If the size of the pixels is too large, small objects will not be detected. Aliasing is a special case of this fault, where the resolution is low enough that artefacts appear to be present in the data that do not exist in the real world. In the object detection example, under-sampling a striped object could result in moiré patterns in the image, causing the system to detect a different shaped object.

Data may also contain ambiguity. A simple example is *lack of precision*. For the purpose of the taxonomy presented in this report, precision refers to the value being measured, rather than the sampling resolution. In the example above, precision would relate to the veracity with which each pixel could represent colour or greyscale values, and would be dependent on the number of bits used to represent the value. Using the analogy of a shooting target, precision is the tightness of a grouping, accuracy is how far the grouping is from the centre of the target, and resolution concerns the spacing of the targets being shot at.

More complicated ambiguities arise when a data value is capable of *multiple interpretations*, e.g. because a word in natural language has two meanings. A common area of ambiguity is in representation of numbers in human-prepared data. e.g. the number "10" could mean two in binary, ten in decimal, or sixteen in hexadecimal. The precision of the number is also ambiguous, and could be interpreted in decimal as 10 exactly, 10 ± 5, or 10 ± 0.5, depending on what assumption was made about rounding. A data fault will arise if the user's interpretation is different to the preparer's.

Faults may also occur when references that are believed to be unique, such as addresses or sequence numbers, turn out to map to multiple objects. In this case, data that may originally have been valid can develop a fault, through *violation of uniqueness*.

Data may be correct, but inconsistent with other data. If a piece of information is repeatedly sampled, or sampled by different mechanisms, there may be *inconsistency between instances of the same data*. This would be self evident from the difference in data values in multiple versions of the same data element. Such an inconsistency could be resolved by a number of methods, including voting, averaging, choosing the data with the best provenance, or asking for the data to be resent.

There may also be *inconsistency between successive data items*, based on the properties of the physical object represented by the data. Examples include cumulative values, such as distance travelled, that are never expected to reduce, or values where the rate of change is limited by physics.

More complicated *inconsistencies between different data items* may arise, based on the known design of the system or data model. These can include mutually exclusive options, or disagreement between metadata and the main data.

While the presence of *inconsistency* is a data fault, it can be useful, in that it is by definition detectable, and can be used to allow other types of data fault to be detected. Careful design can ensure that many value faults will also cause inconsistencies that can be checked for by the system processing the data and dealt with appropriately. However, such checking requires resources at runtime and may not be feasible, depending on the complexity of the calculations involved and the required rate of data throughput. In some cases, it may be possible to adjust the format or content of data structures to make them easier to check, by moving some of the processing burden from runtime to the data preparation process.

Data may be unexpectedly included in a data set (*commission*) or left out (*omission*). Either case may result in unexpected behaviour of the system processing the data, such as misinterpretation of subsequent data. Additional data may result in buffer overruns and corruption of data stored internally to the system processing it. Omitted data could mean that a value relied on by the system is not available. Special cases of these faults are possible where the omitted or added data elements are not complete. These faults are included in the taxonomy as *incomplete data* and *data overrun* respectively. A further special case of commission is *repetition*, where a desired data element is unnecessarily repeated in the data set.

6.5.2. Format

The second high-level category of data fault is *format*. Faults in the formatting of data may not affect the correctness of the content of data, but may mean that a system is unable to understand or process the data. This can be hazardous if the fault is unexpected, however formatting faults can be useful revealing problems with the preparation of data, or subsequent corruption. Formatting is used to impose rules onto the way information is represented, and limit the ways in which it is interpreted. These rules can form the basis of consistency checks, allowing value errors that would otherwise be subtle or plausible to become detectable.

An important aspect of format concerns how quantities are represented. The format needs to specify (or specify a means of identifying) what *units* a quantity has been measured in, e.g. metres vs. feet, to avoid mis-interpretation. It also needs to specify what *datum* the measurement uses as a reference. In some cases this may be included in the definition of the units (e.g. Kelvin versus degrees Celsius), but in others it may not be so obvious. The latitude and longitude of a geographical position within the UK may vary by over a hundred metres depending on whether the datum used is OSGB36 or WGS84.

Closely related to units is *scaling*. This refers to the mapping between the representation and a physical quantity, when the data does not represent the quantity directly. An example would be data representing the output from an analogue-to-digital converter. The output can be represented as a number, but the number must be scaled to determine the actual value being measured. Again, the format should either specify what scaling should be used, or how the scaling should be identified.

Another area that is important to correct interpretation of data is *type*. This concerns the representation of the value. E.g. a sequence of eight binary bits may represent eight individual binary values, an unsigned number between 0 and 255, a signed number between -128 and 127, or various other possibilities. Variations in data types also include the differences between integer types and floating point types with various levels of precision, and endianness (concerning the ordering of bits or bytes within the data element). Type faults lead to data that was written as one type being interpreted as another, causing the wrong value to be obtained.

The chosen data type may be able to represent more values than is appropriate for the information being represented (e.g. a temperature below -273.15 degrees Celsius is invalid). The format should specify the range of acceptable values, allowing *data out of range* faults to be detected.

Not all data represents quantities. Format also needs to specify how non-numeric values should be represented and interpreted. If data includes free text, the language should be specified, to ensure that the system can use the data. e.g. "colour" (en-gb) and "color" (en-us) may have the same meaning to a human, but the difference in spelling between American and British English could cause a software-based system to reject the value. It may be necessary to restrict the allowable values to a limited vocabulary. *Language* faults encompass failures of elements of the data to comply with the allowable vocabulary, whether that is defined by a human language, a machine language (e.g. Ada or the eXtensible Markup Language (XML)), or a subset specific to the particular format. They are the non-numerical equivalent of out of range faults.

Grammar faults concern the way that values are arranged in data, including syntax, punctuation, and structure. The grammar of a format will determine how a system can distinguish between different data elements. The grammar rules for a format may include some elements of sequence, however, data can also be *out of sequence* while following correct grammar. This type of fault might arise either when a meaning is derived from the sequence of data elements (e.g. when it is assumed that a sequence of measurements appear in chronological order), or where the data represents intent or builds on previous data vales, and it is important that elements are interpreted in a particular sequence.

6.5.3. Timing

Timing faults relate to the period when data is required for use. Many timing faults are analogous to faults in the domain of meaning, especially *omission* and *commission*. In the time domain these relate to the presence or absence of bulk data (sets, packets or streams), as opposed to data elements within a set. These faults are generally applicable to data that is communicated, rather than data that is held statically within a system.

Omission faults can be divided into *availability*, where the source that is supposed to provide the data is not available; *existence*, where the source is available, but does not hold the required data; access, where the source has the data, but will not pass it to the recipient; and loss, where the data is sent from the source, but does not reach the recipient. Loss may also occur after receipt of data. These faults may have a number of causes, depending on context, but will all result in required data being unavailable for processing and use at the time when it is required.

Commission faults cover cases where extra data is received unexpectedly. A special case is *repeated receipt* of duplicate data packets. Commission faults could result in systems using extra processing or storage resources to deal with the unexpected data, or taking unwanted or nugatory actions if the duplication was not detected.

Data may be *early* or *late*, either with respect to the time that it is required for use, or the time that it was prepared. Data that arrives early for use may be discarded, or be stale by the time it is used. Data that is prepared early or late may not properly represent the intended object. Data that arrives infinitely late can be considered an omission. A special case of late data is when data is used beyond its period of *validity*. In this case it is likely that a value fault will occur, as the information represented by the data is likely to have changed. It could be argued that expired validity is early data (i.e. that it was prepared too early to be valid at the time of use), but the author has taken the view that either the data is so late that it has expired before it is available for use, or that an update to the data should have been made available, and was late.

When some data is early, and other data late, an *out of sequence* fault occurs. This may result in the same problems as the value domain out-of-sequence fault, unless a way of re-assembling data packets into the correct order is provided.

6.5.4. Provenance

In addition to having data with the correct meaning and format, at the correct time, it is important that the data has the desired *provenance*. Provenance gives confidence that data has been prepared properly and processed, stored and communicated in a way that has preserved its key characteristics, such as format, meaning and timeliness. A provenance fault occurs when there is not a sufficient process argument to give assurance about the integrity of the data through each stage of the supply chain from preparation to use. A special case of this fault is when there is no traceability, and hence there is an *unknown provenance*.

Provenance faults can also occur when data that appears to be from one source is actually from another (*masquerade*). This can occur due to a deliberate attack, or accidentally, e.g. when test or training data is inadvertently injected into a system.

6.6. Use of the Data Fault Taxonomy

The taxonomy proposed in section 6.4 is intended to be used as a checklist to aid hazard identification. In safety analysis, it is anticipated that a system designer or safety analyst would use it as input to an investigation of what failings in data could lead to unsafe behaviour at the system level. It can also be used to consolidate multiple known causes of data issues into groups that can be independently managed.

The high-level taxonomy categories (meaning, format, timing and provenance) would be used as prompts or guide words during a structured walk through of the data structures used in a system. When a high-level category had been found to be relevant, the lower-level guide words within that category would be used, in order to draw out more specific forms of potential fault.

Figure 2 illustrates that a fault may have many causes, and give rise to many hazards at the system level. While the different causes and system-level hazards could be independently managed through various different mitigation measures, the fault provides a concept at the right level to facilitate simple understanding and management of the problem.

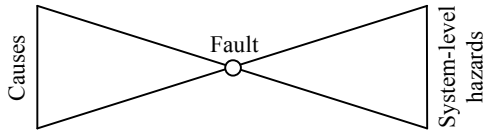

Figure 2. Bow-tie diagram showing relationship of causes to faults and hazards.

The taxonomy could also be used by someone specifying the requirements or coding the implementation of a data format or a system that handled data. It would give a series of prompts as to the types of failure mode that the system should be required to handle. Similarly, it could aid test case design.

While a single check-list could not hope to identify every specific cause of data-related problems, it is hoped that the high-level categories presented by the taxonomy are sufficiently generic to encompass all potential faults.

7. Data-Related Accidents

In order to validate the new taxonomy of data faults, the author has reviewed accident and incident reports from a number of sources. This section of the report summarises those accidents and incidents that the author considered had data-related factors, either in their cause, or how the subsequent sequence of events developed.

It should be noted that for brevity, the précis presented here are greatly simplified, with an emphasis on highlighting the contribution made by data. Most other contributary factors are likely to have been omitted from the description, and the reader should make reference to the source documents for a fuller understanding of the causes of each event.

For each event, the data faults identified have been classified according to the new taxonomy, with the classification highlighted in **bold text**. This exercise has sought to show that all relevent contributions can be satisfactorarily classified according to the taxonomy, and conversely to show that faults in the taxonomy do actually manifest themselves in the real world.

7.1. Air Accident Investigation Branch Reports

All the précis in this section are based on reports published on the Air Accident Investigation Branch (AAIB) website [6]. The author reviewed all 31 full investigation reports published by the AAIB from 2001 to the end of August 2008. Only the two believed to have some relevance to data considerations are précised here.

7.1.1. F-OJHI, Birmingham International Airport, UK

Ref: AAIB report 7/2007

On 23 February 2006 an Airbus A310-304 descended early on a Localiser/DME approach to Runway 33. The crew aborted the descent at 164 ft above ground level when they received a Ground Proximity Warning System (GPWS) alert. The aircraft went around and started a second early descent, until being guided onto the correct course by the air traffic controller. The incident occurred because the wrong Distance Measuring Equipment (DME) setting was used.

This came about because the Flight Management Computer was using a database that did not contain the relevant approach path (**omission**). The crew were required to manually set up the VHF Omni-Range (VOR) system to use the DME beacon on Runway 33. They did not do this, and the system autotuned to a different DME beacon. This meant that there was an **association** fault concerning the DME tuning. As a result, the distance reported by the DME was being measured from the **wrong datum**, causing the aircraft to descent early.

7.1.2. G-MEDG, Khartoum Airport, Sudan

Ref: AAIB report 5/2007

An Airbus A321-231 was on its final approach to Runway 36 at Khartoum Airport on 11 March 2005 when the crew attempted to conduct a Managed Non-Precision Approach using the autopilot. Darkness and wind-blown sand meant that visibility was restricted. The autopilot used data from the aircraft's Flight Management and Guidance System navigation database, but the crew were using a chart produced by a different supplier, on which the descent point for the approach differed by 0.6 nm. This caused them to believe that the aircraft was too high for the approach. The pilots overrode the autopilot, then became confused, descending too fast and necessitating an emergency go-around. The aircraft avoided the terrain by 121 ft.

The incorrect location on the chart was an **inaccuracy**, and the difference between that data and the navigation database an **inconsistency between different data items**. This inconsistency arose partly because the data supplier had not received a relevant update from the Sudanese authorities, which had been received by the navigation database supplier (**late** data or **omission**, depending on whether the update was eventually received). The earlier data that the chart was based on had also contained **inconsistency between different data items**

representing the vertical and horizontal approach paths. Although the supplier had detected this inconsistency, the method they had used to resolve the problem had led to **inaccuracy** in the published chart.

7.2. Marine Accident Investigation Branch Reports

Ships have long made use of data in the form of charts for navigation. They are now increasingly using electronic data, in navigation, communications, and to co-ordinate operations such as cargo loading and movement.

All the précis in this section are based on reports published on the Marine Accident Investigation Branch (MAIB) website [7]. The author reviewed all 71 full investigation reports published by the MAIB from 2006 to the end of August 2008. Only those believed to have some relevance to data considerations are précised here. An additional report (4/1998) was included as it was already known to have relevant aspects.

7.2.1. Westhaven

Ref: MAIB report 4/1998

On 10 March 1997 fishing boat called *Westhaven AH190* caught her trawling equipment on a seabed pipeline, capsized and sank. The rescue services were alerted by an Emergency Position Indicating Radio Beacon (EPIRB), the signal from which was picked up by satellite. Unfortunately, the EPIRB was registered to a different fishing vessel, called *Westhaven FR375*, which was contacted and found to be safe. The search was called off, but it later became clear that the correct EPIRB had not been found, and there might still be a vessel in distress.

Westhaven AH190 had originally been registered under the port number FR375, but was sold and re-registered as AH190, but retaining the same name. The previous owner bought a new boat and registered it under the same name and number as his old vessel. The details in the EPIRB registration database were not updated, causing an **association** fault: the name, port number and owner details in the database were still all consistent with each other, but they no longer referred to the boat that the EPIRB was actually fitted to.

7.2.2. Lykes Voyager / Washington Senator

Ref: MAIB report 4/2006

On 8 April 2005 the container ships *Lykes Voyager* and *Washington Senator* collided in the Taiwan Straights, after making evasive manoeuvres which actually put them on a collision course. The master of the *Washington Senator* had attempted to contact the *Lykes Voyager* by VHF radio, to negotiate how the vessel would pass. However, no identification data was passed during the communication, and the arrangement was actually made with some other (unknown) ship nearby. By the time that the *Washington Senator* realised that the other ship was not manoeuvring as expected, it was too late to avoid the collision.

This was a **provenance** data fault, as data believed to be from one source actually came from another. If the data had been labelled as coming from the correct ship (e.g. the wrong callsign was used), then it could have been classified as a **masquerade**.

7.2.3. MV Lerrix

Ref: MAIB report 14/2006

On 10 October 2005, *MV Lerrix* grounded in the Baltic Sea, after the master fell asleep on the bridge of his ship. Although not causal to the accident, it was later found that the master had been navigating using his own private laptop and GPS receiver, running pirated navigation software. The data used by this software was five years out of date. Using the software may have given the master a false sense of security, as it did not integrate inputs

from the ship's radar or Automatic Identification System (AIS) receiver, or provide alerts for navigational hazards such as deviation from course or proximity of land.

The immediate data fault here is **validity exceeded**, as the navigational data had not been updated (the UK Hydrographic Office publishes updates weekly in its Admiralty Notices to Mariners series), although this was not a causal factor. It is however possible that the fact that the ship's master believed he was using highly accurate GPS data for navigation allowed him to become overconfident and cease checking the data from his other navigational instruments. Together with the comforting glow of his laptop screen on the dark bridge, this may have contributed to him falling asleep. While this is more to do with the master's use of data than the data itself, it could be seen as a **provenance** fault, as more trust was placed in the navigation data than was merited by its source.

7.2.4. Dieppe

Ref: MAIB report 18/2006

On 5 December 2005 the passenger ferry *Dieppe* grounded in the entrance to the port of Newhaven. She was being navigated using GPS and electronic charts and ran aground in an area where the charts showed that she should have had a sufficient depth of water. Recent bad weather had caused the channel to silt up, reducing the available depth by 1.3 m since the previous survey (7 days earlier). The vessel was running late, compounding the problem as the tide was falling. It was not using its echo sounder. Although the *Dieppe*'s master was aware of the hazard of silting, he had not been passed a recent risk assessment which gave guidance on conditions for entry to the port, and did not take any further precautions himself.

The first fault here is a value fault: the chart was **inaccurate** as it showed a greater depth of water than was present, due to the world changing since the data was prepared. There were also two **late** data faults, since the channel was not re-surveyed quickly enough after the heavy weather and the master was not provided with the relevant risk assessment in time to prevent the accident.

7.2.5. P&O Nedlloyd Genoa

Ref: MAIB report 20/2006

The container ship *P&O Nedlloyd Genoa* encountered high seas in the North Atlantic, losing 27 ISO containers overboard and having a further 28 crushed on deck. The cargo had been loaded according to a flawed loading plan which exceeded the maximum allowable weight for some stacks of containers, and violated a rule that heavier containers should not be stacked on lighter ones. These flaws in the planning data were not detected either by the planning software, or the ship's stability computer. Partly this was because the software did not distinguish between different height containers and therefore applied the wrong rule set when checking stack weights.

Additionally, some of the remaining damaged containers were found to have actual weights that varied from their declared weights by up to 20%. These deviations of the actual cargo from that represented by the planning data contributed further to the overloading of the cargo. The overloading, combined with inconsistent lashing arrangements and high accelerations due to the ship rolling in heavy weather, meant that some of the lower containers were crushed, causing their stacks to topple.

This accident can be classified as **inconsistency between different data items**, as the loading plan was not consistent with the data provided in the ship's stability book. **Inaccuracte data** was also present in the records for individual containers.

7.2.6. FV Harvest Hope

Ref: MAIB report 21/2006

The fishing vessel *Harvest Hope* foundered on 28 August 2005 after snagging its trawl gear on seabed debris in the vicinity of a number of North Sea hydrocarbon pipelines. Because of the possibility that one of the pipelines might have been snagged or damaged by the wreck, it was advisable for the pipeline operators to cease production or reduce the pressure in the pipelines until the extent of any damage could be surveyed. However, the largest scale Admiralty chart for the area only showed two of the four pipelines present. This meant that the Maritime Rescue Co-ordinating Centre (MRCC) contacted initially only two of the three relevant pipeline operators to warn them of the incident. Had one of the other pipelines been damaged, this could have increased the severity of any pollution incident.

Although the Hydrographic Office had been notified of the new pipe, their policy was only to make amendments to charts when a new route was for some reason "safety-critical", or when the chart scale is large enough to show the gap between new and existing pipelines. In this case the gap would have been only around half a millimetre on the 1:200 000 scale chart.

The MRCC had a database of subsea hazards available that did identify all the relevant pipelines. It was not used however, because of a perception that it was less accurate than Admiralty Charts, as it was only updated twice a year, rather than as and when changes were found necessary.

The failure of the Admiralty chart to identify all the artefacts of interest in the area can be classified as an **insufficient resolution** fault.

7.2.7. FV Brothers

Ref: MAIB report 1/2007

On 1 June 2006 the fishing vessel *Brothers* grounded and sank off Scotland with the loss of both its crewmen. The accident was probably caused by the crew falling asleep due to the effects of fatigue and alchohol. While data was not a causal factor in the accident, the rescue operation was delayed by a data error which could have influenced the crew's chance of survival.

Once the vessel had been reported missing, the MRCC attempted to use mobile phone network records to determine the last time the vessel was known to be afloat, and use this information together with their assumed course and tidal patterns to calculate a search area. Although they were passed a correct time for the last contact with the skipper's phone, they used a value approximately an hour earlier in their calculations. This meant that rescue efforts were initially concentrated in the wrong search area, affecting the crew's chance of being found alive.

This was an **inaccurate value** data fault.

7.2.8. Arctic Ocean / Maritime Lady

Ref: MAIB report 2/2007

On 5 December 2006 the container ship *Arctic Ocean* collided with the cargo vessel *Maritime Lady* as it entered the Elbe river from Brunsbüttel Lock. Possible contributory factors include potentially misleading information in the Admiralty Sailing Directions on the rules for right of way in the river, and the fact that the Vessel Traffic Service controlling the area did not provide *Arctic Ocean* with a traffic report before she started the manoeuvre.

The issue with the right of way rules can be classified as **ambiguity**. As the traffic report was not provided at all, this factor is classified as **omission** in the time domain.

7.2.9. Hilli

Ref: MAIB report 4/2007

On 10 October 2003 an explosion occurred in the boiler of the liquid natural gas carrier *Hilli*. Two contractors had been cleaning the boiler using a sulphamic acid-based chemical cleaning agent. The acid reacted with the steel wall of the boiler to produce hydrogen gas, despite the presence of an inhibitor that was supposed to prevent this. The hydrogen was ignited by a halogen lamp, causing the explosion. One of the men received fatal injuries.

A potential factor in the accident is that the product data sheet and safety data sheet for the cleaning agent did not describe the function of the inhibitor, the possible evolution of hydrogen, or the risk of the explosive atmosphere that could result. This meant that sufficient precautions were not taken by the contractors.

The lack of data about the hazardous effects of the cleaning agent in its data sheet can be classified as an **omission** in the value domain.

7.2.10. MV Thunder

Ref: MAIB report 12/2007

MV Thunder dragged its anchor and grounded off Mostyn in the Dee Estuary approaches on 10 August 2006, after the vessel had anchored in an inappropriate area. The *Thunder*'s master had not been provided with a sufficiently large scale chart for navigation to the port of Mostyn and knew that he would be unaware of potential grounding hazards nearby. He planned to anchor in a safe location offshore to await boarding by a pilot. However, he was persuaded to anchor further in towards the port by an agent of the shipping company, who had emailed the GPS coordinates of a "preferred anchorage" and a list of GPS waypoints to allow navigation there. The master had assumed that this information had come from a qualified pilot or harbourmaster, which was not the case. The master's copy of the relevant Admiralty Sailing Directions had also not been amended with the most recent changes concerning Mostyn pilot boarding locations. This meant he did not notice that the new location was not a recognised boarding point. The scale of map being used made it difficult for the crew to identify that the anchor was being dragged when strong winds developed overnight.

The use of the inappropriate chart is an **insufficient resolution** fault, while the fact that the proper chart was not available is **omission**. The fact that the Sailing Directions had not been properly updated means that there was also a **validity exceeded** fault. Finally, the reliance on navigation data from the company agent was a **provenance** fault, as the master assumed that it had been supplied by a qualified source.

7.2.11. Octopus / Harald

Ref: MAIB report 18/2007

The jack-up barge *Octopus* was being towed by the tug *Harald* in Stronsay Firth, off the Orkney Islands on 8 September 2006, when it grounded on an uncharted bank. Its legs had been lowered to a depth of 13m for stability in towing, but the depth of water over the bank was only 8m. The tug was using the best available chart of the area, which showed that at least 20m of water could have been expected.

While the chart was the latest available, it was based on lead line survey data from the mid 1840s. This data is likely to have been accurate, but without sufficient resolution to highlight the hazard of the shallow area (i.e. the soundings were not close enough together). It is also possible that the bank might have either formed or shifted during the intervening 160 years.

The tug had used a paper version of the chart for planning its voyage and an electronic version for the voyage. While the paper version contained warnings about the quality of the source data, the electronic chart did not contain any information about the level of confidence in the data. This should have been provided as Category of Zone of Confidence (CATZOC) metadata.

The fact that the chart did not highlight the bank is probably an **insufficient resolution** fault, but could also be an **inaccuracy** if there was an inaccurate data point coincident with the location of the bank. The lack of traceability between the information displayed on the electronic chart and the source data is a **provenance** fault, which could have led to more confidence being placed in the chart than was deserved.

7.2.12. Annabella

Ref: MAIB report 21/2007

The container ship *Annabella* encountered heavy seas on 25 February 2008, resulting in the collapse of a stack of seven cargo containers, damaging the containers (some of which carried hazardous cargo) and the hold of the ship. Stresses caused by the movement of the ship resulted in the lowest containers being crushed, as the stack had been piled too high both for the particular hold location and the stacking limits of the containers. The stack of 30 ft (non-standard size) ISO containers weighed 225 tonnes, whereas the limit for that area of the hold was 150 tonnes, and the limit for some of the lower containers was 100 tonnes.

The loading plan from the ship was generated by the cargo company's shore-based planners using software that was supposed to embody the stability and stowage data provided by the vessel's manufacturers. However, this software silently converted the dimensions of the 30 ft containers to 40 ft, meaning that the wrong stack limits were used for checking. The loading plan data was passed to planners at the shipping terminal, which checked the availability of the containers, but not their stability. It was then input into the loading computer onboard *Annabella*, which again did not recognise the 30 ft containers, and applied 40 ft limits. In neither case did the data used by the loading software reflect the full detail of the ship's loading manual.

Additionally, the loading manual had been devised for a particular configuration of the vessel. At the time of the accident, the ship was configured with a different metacentric height, due to extra ballast being used during ice operations. This should have resulted in lower limits being applied, as greater forces would have been experienced by the cargo.

Although the loading limits for each container was recorded on the container itself, it was not present in the loading data. Some of the lower containers had limits below the ISO standard of 213 tonnes, but there was no check to ensure that this was not exceeded.

This accident involved **value** faults in the internal representation of the container dimensions, which resulted in **inconsistency between different data items** as the loading plan did not match up to the ship's stability book data. It could also be argued that there was **omission** of the loading limit information from the loading plan. However, this was a flaw in the design of the system, rather than the data itself.

7.2.13. M-Notices

It is interesting to note that due to accidents similar to some of those mentioned above, the UK Maritime and Coastguard Agency has published various circulars (M-notices) to warn mariners about specific data-related hazards. These include advice on the mandatory requirements for registration of Emergency Position Indicating Radio Beacons (EPIRBs) in a central database, keeping the registration data up to date, maintaining the equipment and, where necessary, frequently updating the location information in the beacon (MSN 1810, MSN 1816, MGN 302); the importance of correctly setting Automatic Identification System data (MIN 321), and not relying on that data or VHF radio for collision avoidance (MGN 324); and only using Electronic Chart Display and Information Systems that meet international standards for display of relevant features and protection of data (MIN 316) [32].

7.3. Rail Accident Investigation Branch Reports

Modern railways in the UK use a number of data-driven or information processing systems. The Total Operations Processing System (TOPS) system is used to plan and record information of trains, including the length, weight and order of their component wagons. Train movements and signalling must also be coordinated

with timetable data and data about the layout of the railway. Most trains themselves are fitted with electronic communications and control gear, as well as On Train Data Recorders (OTDRs).

All the précis in this section are based on reports published on the Rail Accident Investigation Branch (RAIB) website [8]. The author reviewed all 90 full investigation reports published by the RAIB from 2006 to the end of August 2008. Only those believed to have some relevance to data considerations are précised here.

7.3.1. RAIB Autumn Adhesion Investigation

Ref: RAIB report 25 (Part 3)/2006

In 2007, the RAIB published a comprehensive investigation into a number of adhesion incidents that occurred during the autumn of 2005. These included a number of instances where trains slid or were unable to halt as quickly as expected, causing them to pass signals at danger and overshoot stations. Several near misses could also have led to collisions. The lack of adhesion was caused by factors such as icing, precipitation and the presence of contaminants on the rail head (such as leaf residue).

Much of the mainline fleet of rolling stock is fitted with software controlled wheelslide prevention (WSP) and sanding equipment. The investigation identified that many of the systems involved had not been optimally configured to respond to the prevailing track conditions. Partly this was due to the choice of strategy in the relevant standards. However the lack of involvement of some of the rolling stock operators in specification of parameters was also identified as a contributory factor.

Poor choice of set-up parameters for the WSP software can be classified as an **inaccurate value** data fault.

7.3.2. Despatch of unsecured load from Besford Hall

Ref: RAIB report 6/2006

On 21 February 2006 a train was despatched from the Besford Hall yard in Crewe with an unsecured load. Due to a breakdown in communications between a shunter and the yard's team leader, the train was assembled using the wrong trucks. These trucks had not been prepared and could at any point have caused an accident by shedding their load.

The train should have been assembled according to data provided by the Total Operations Processing System (TOPS) system. This specified which trucks should have been used, in which order. However, the shunter did not check the truck identification numbers against the data.

The initial fault here was not a data fault. Human factors issues meant that the shunter did not gain the correct understanding of their task. However, this resulted in an **association** fault. Once the train had been assembled, the data in TOPS, which was correct for the intended train, did not match the actual train that passed over the network. The incorrect data from TOPS could potentially have then be used to make safety-related decisions, such as a choice of route that would satisfy the restrictions applicable to particular units of rolling stock.

7.3.3. Unauthorised train movement at High Street Kensington

Ref: RAIB report 19/2007

On 29 April 2006 a London Underground train on the District Line went the wrong way at a set of points. The driver realised the problem and, despite communications problems caused by poor reception and flat radio batteries, contacted the service controllers, who authorised him to reverse the train, to allow the points to be reset for the correct route. The train backed up further than was authorised, passing a stop signal and causing the traction current to be discharged, halting the train. The driver had to exit the cab to determine where the train had stopped and in doing so fell, injuring his knee. The incident caused delays of over an hour on the line, and the injured driver was off work for eight days.

The initiating event was the selection of the wrong route for the train by the *programme machine* controlling the points. This is an electromechanical machine which uses holes punched into plastic tape to set the correct route for each train. A temporary programme tape was being used due to maintenance work being carried out nearby, and it is suspected to that this tape had been incorrectly punched. In other circumstances, a similar fault could have resulted in a collision or derailment.

The initial fault with the punched tape may be classified as a **value** fault that gave incorrect intent data. The subsequent failures of communication can be seen as an **availability** fault, where dynamically required data was unavailable due to failure of the data source.

7.3.4. Freight Train Derailment at Maltby North

Ref: RAIB report 24/2007

On 28 June 2006 a Freightliner coal train was derailed at Maltby North when points changed under the train. The control tables for the points required that if a train was present on the section of track approaching the points, it must have been there at least two minutes before the points could be opened (to give the train time to either pass or halt at the relevant signal).

One of the causal factors of the accident was that this rule was not actually implemented by the electro-mechanical mechanism controlling the points.

This accident would not generally be regarded as a data fault, as the contributary factor in question relates to a relay-based electrical implementation. However, as the connection of the relays could be seen as a representation of the intent information given in the points control table, the definition could be stretched. The fault would then be classified as **incomplete data**, as a required element was not represented.

7.3.5. Possession Irregularity near Manor Park

Ref: RAIB report 26/2007

On 19 March 2007 a crew of Kier Rail maintenance workers working on a line near Manor Park station was approached by a train travelling at around 80 mph. Although the workers jumped clear the train hit their wheelbarrows and two of the men were injured by flying debris. The men were on the track because they believed that their work site was in an authorised “engineering possession”; a section of track that had been protected from rail traffic to allow work to take place.

Network Rail operates a Possession Planning System, which generates a Weekly Operating Notice (WON) detailing all approved possessions. Contractors working on the railway negotiate the possessions they will use at planning meetings held 14 and 6 weeks prior to the time of the possession. In this case, Network Rail had changed their plans regarding the timing and limits of the possession that Kier Rail had been going to use, necessitating Kier Rail to change their plans and work on a different site instead. Although plans were drawn up for the new site, communications failures meant that the new possessions data from the Network Rail systems was never reflected in Kier Rail’s internal possession planning spreadsheet, and consequently the work crew was sent to the wrong work site. The crew did not read the plan for their work and thus did not realise that it had been written for a different location. The site supervisor’s workload meant that he was not physically present at the site and, in permitting the work to start by mobile phone, never realised that the crew was not at the correct site.

This fault is an **inconsistency between different data items**, as different plans came into existence that were not updated in step with each other.

7.3.6. Passenger door open on a moving train near Desborough

Ref: RAIB report 31/2007

A passenger door on a Class 222 train Midland Mainline train manufactured by Bombardier Transportation opened while the train was moving, on 10 June 2006. The microswitch which reported the status of the door lock had developed a fault. This was detected by the Train Management System (TMS), which took the door out of use but did not report the problem to the crew. The lock itself had not properly engaged, and repeated cycles of the inflatable door seal being deployed meant that the door gradually edged open.

The reason for the door lock not engaging is believed to be an unintended change to the software parameter for the door motor limit current. This restricted the closing force and meant that when the microswitch fault was detected and the TMS cut power to the door motor, the lock mechanism had not fully engaged. Unlocked, and with no positive closing force, the door gradually worked open.

This is an example of an **inaccurate** value fault, caused by the value changing after preparation.

7.3.7. Fire on HGV shuttle in the Channel Tunnel

Ref: RAIB report 37/2007

On 21 August 2006, a fire broke out on an Heavy Goods Vehicle (HGV) on board a shuttle train in the Channel Tunnel. All passengers were safely evacuated but the HGV was destroyed and there was some damage to the carrier truck and the tunnel infrastructure. During the emergency response to the incident, a delay was incurred in summoning the UK local authority emergency services (as opposed to the tunnel's own emergency services), as they required a postcode to locate the incident. The incident location (under the sea, about 11 km from the English coast) did not have a postcode, and the tunnel's Fire Detection Controller took 5 minutes to establish a suitable address to allow the tasking. Had the fire been more serious, this delay could have resulted in the reduction of the chances of survival of any injured persons.

This issue can be classified as an **existence** fault, as the data required by the emergency services was not available in the required format.

7.3.8. Derailment at Cromore, Northern Ireland

Ref: RAIB report 42/2007

On 14 April 2007 an ultrasonic test truck derailed near Cromore. The short wheelbase of the truck, combined with the speed at which it was being towed and the profile of the track had led it to bounce and oscillate until it derailed. The main causal factors were that the Weekly Operating Notice (WON) giving details of special train movements that week contained the wrong speed limit values for the type of track and the Special Operating Instruction (SOI) for the specific train movement, which should have confirmed limitations for the train, had not been issued. The WON had been prepared using data from an engineer's memory, rather than the original vehicle approval or previous SOIs, and a new SOI was not issued because a staff member was on leave.

Several data faults can be discerned in this accident: a fault in **availability** led to an **inaccuracy**, as data from memory (a source without the appropriate **provenance**) was substituted for the properly prepared version.

7.3.9. Derailment at Duddeston Junction, Birmingham

Ref: RAIB report 16/2008

On 10 August 2007 several wagons of a Freightliner train derailed near Birmingham, spilling one of the containers onto the track and blocking several lines. The accident was caused by a combination of a track

twisting fault and the uneven load configuration of one of the wagons. While data considerations were not causal to the accident, use of data could have helped avoid it occurring.

As containers were received by the freight terminal, data about them was input into a computer system called ERIC. This data included the size and weight of each container. After the containers are loaded onto the train, their positions are also entered into ERIC, and data from ERIC is fed into the Total Operations Processing System (TOPS). TOPS also contains information about which wagons form the train, and the limitations on those wagons. Had cross-checks been built into ERIC or TOPS, the system could have used the data that was already available to identify that one of the wagons was loaded in a configuration that was outside its allowed limits. This would have been classified as an **inconsistency between different data items**.

7.4. Other Selected Accident Reports

7.4.1. ZK-NZP, Ross Island, Antarctica

Ref: New Zealand Office of Air Accidents Investigation Report 79-139 [11]

On 28 November 1979 Air New Zealand Flight 901, a McDonnell Douglas DC-10-30 on an Antarctic sightseeing flight, crashed into the slopes of Mount Erebus on Ross Island in Antarctica with the loss of all 257 souls on board. The planned flight path did pass directly over Mount Erebus but had been incorrectly programmed into the flight management computer that guided the aircraft, until immediately prior to the accident flight. The plan had held an incorrect location for a turning point near the mountain, McMurdo Station. This incorrect course had taken previous flights to the side of the mountain, rather than directly over it, and the aircraft had been able to descend to allow the tourists a better view. Before the fatal flight, ground crew had reprogrammed the computer with the correct route data, without informing the pilots. It appears that the pilots encountered low visibility and believing that they were over sea-level ice, descended to escape the cloud. The visual conditions seem to have meant that the crew could not distinguish the uniform, snow-covered ice slope from the cloud-covered sky.

The original mis-programming of the flight computer is a case of an **inaccuracy**. The fact that this data was then changed is an **inconsistency between instances of the same data**: the aircraft was not programmed for the same route as on previous occasions, leading the pilots to make the wrong assumptions about the terrain they were flying over.

7.4.2. N651AA, Cali, Columbia

On 20 December 1995, American Airlines Flight 965, a Boeing 757-223, was on a VOR DME approach to Alfonso Bonilla Aragón Airport in Cali, Colombia, when it crashed into a nearby mountain at Buga. The crew had attempted to program the flight management computer to fly towards a DME beacon called Rozo, by trying to select the identifier shown for it on their charts, “R”. However, the beacon was identified as “ROZO” in the navigation database, which used “R” to identify a beacon called Romeo, over 130 km away. The plane commenced a turn towards Romeo, which caused it to fly into nearby high terrain [40].

While beacon designations are supposed to be unique within a country, the Colombian government had assigned the designation “R” to both Rozo and Romeo. This was a **violation of uniqueness**. When the navigation database was prepared, the supplier acted to preserve the integrity of the database by maintaining unique entries, and designated Rozo by its full name. However, this created an **inconsistency between different data items** in the database and published navigation charts. This led to the wrong choice of identifier by the pilots, and an **association** fault, when the pilots believed that data relating to one beacon actually related to another.

Gibbon and Ladkin also identified lexical ambiguities in the data exchanged between the pilots and air traffic controller, where the term “Rozo” had **multiple interpretations** and could refer both to a navigation beacon and an approach route [24].

7.4.3. N52AW, near Pasamayo, Peru

An Aeroperú Boeing 757-23A crashed into the Pacific Ocean off the coast of Peru on 2 October 1996, after the crew started to receive erratic information from onboard instruments. The crew started to return to Lima Airport using manual controls but, while believing the plane to be at a safe altitude, crashed into the sea. The accident appears to have been initiated by pitot tubes having been covered for maintenance and not unblocked before take-off, causing the sensors that provide altitude and air speed information to malfunction. When the plane neared the airport, the pilot asked air traffic control to confirm its height, and was told 9000 ft. The aircraft had actually been travelling at 1500 ft, but the data available to the control tower was provided by a transponder on the aircraft, and hence based on the same flawed data as the aircraft's other systems. Believing the data provided by the tower, the pilot started his descent too soon and crashed into the water [30].

The blocked pitot tubes meant that pressure transducers were not exposed to the expected air pressure, and hence the calculated air speed and altitude of the aircraft were **inaccurate**. Because of the inaccuracies, some of the data derived from these values was **inconsistent with other data items** and the pilots were able to detect the problem. The fault that led immediately to the accident was a **masquerade**, as data that appeared to come from an independent source at the air traffic control tower actually came originally from the aircraft. This meant that the crew put an unmerited level of trust in the data.

7.5. Summary of Accident Analysis

The proposed new data fault taxonomy appears to have been successful in classifying the data faults presented in this section. The accident reports that have been analysed covered systems implemented using a wide range of technologies including human and electromechanical systems, as well as the software-based systems more normally associated with data.

While the majority of faults were classified within the *meaning* category, a reasonable number of *timing* and *provenance* faults were also identified, as shown in Table 10. However, it is notable that few faults were identified in the *format* category, and a number of sub-categories were not used.

Serial	Fault Category	No. of occurrences
1.0.0	Meaning	
1.1.0	Value	2
1.1.1	Meaningless	
1.1.2	Inaccurate	11
1.1.3	Association error	4
1.2.0	Insufficient resolution	3
1.2.1	Aliasing	
1.3.0	Ambiguity	1
1.3.1	Multiple interpretation	1
1.3.2	Violation of uniqueness	1
1.3.3	Lack of precision	
1.4.0	Inconsistency	
1.4.1	Between instances of same data	1
1.4.2	Between successive data items	
1.4.3	Between different data items	8
1.5.0	Omission	2
1.5.1	Incomplete data	1
1.6.0	Commission	
1.6.1	Repetition	
1.6.2	Data overrun	
2.0.0	Format	

2.1.0	Wrong scaling or units	
2.2.0	Wrong datum	1
2.3.0	Wrong type	
2.4.0	Data out of range	
2.5.0	Wrong language	
2.6.0	Wrong grammar	
2.7.0	Data out of sequence	
3.0.0	Timing	
3.1.0	Omission	4
3.1.1	Availability	2
3.1.2	Existence	1
3.1.3	Access	
3.1.4	Loss	
3.2.0	Commission	
3.2.1	Repeated receipt of data	
3.3.0	Early	
3.4.0	Late	3
3.4.1	Validity exceeded	2
3.5.0	Data out of sequence	
4.0.0	Provenance	5
4.1.0	Masquerade	2
4.2.0	Unknown provenance	

Table 10. Occurrence of fault types within reviewed accident reports.

Possible reasons for the low proportion of *format* faults could include the following:

- Good design means formatting faults are rare.
- Formatting faults are easy to detect, so are often rectified before they exhibit system-level behaviour.
- Formatting faults are difficult to detect, leave little evidence, or manifest as other types of fault (e.g. value faults or omissions), so are not often identifiable in accident reports.
- Formatting faults are not hazardous, so do not appear often in accident reports.
- The sample of accident reports was not representative.

In the opinion of the author, a combination of the first three bullets is likely. However, further validation with a larger sample of accident and incident data would be required to determine the true reason(s).

While some categories may be rare as factors in accidents, this would not necessarily mean that they would not be useful prompts when used as a checklist for hazard identification or system design. This would have to be verified through practical use of the taxonomy.

The analysis of the accident reports has also raised some points with regard to how faults develop in an accident sequence. The same fault can in many cases be classified differently depending on the scope one looks at: a value fault in data used by a single subsystem can be seen as an inconsistency when compared with other linked subsystems. In many cases more than one data fault is relevant, and faults may cascade through a system. In some of the accidents reviewed, such as the N52AW crash and G-MEDG incident, the original data faults had been identified, but recovery from the faults was not complete and further data faults occurred, leading to the dangerous event.

8. Areas for Further Work

8.1. Development of the Data Fault Taxonomy

The taxonomy presented in this report requires use to further validate it as a workable tool for hazard identification. While using it to classify data faults in accident and incident reports from other countries and technology domains may have some benefit, it would be more productive to give the taxonomy practical use in real hazard identification exercises.

It is likely that such practical use would suggest refinements to the taxonomy. These might include areas where the hierarchy has been over-developed, such that it can be simplified by using only a higher-level category, or conversely where the range of hazardous faults seen in real systems merits more categories, to allow better distinction between types of fault.

Areas where the author is unsure that the taxonomy has reached its full maturity include the following:

- The distinction between the time and value domains for omission and commission faults. Do these need to be kept separate, or could they be better combined, as in SHARD? If they are kept separate, should they be given different labels to emphasise the differences?
- The sub-categories of omission fault in the timing domain. Is it useful to distinguish between faults where the source is not available, the data is not available and the source will not grant access to data, or are they all just causes of the same thing?
- Provenance. Does this category need expanding or splitting up, to distinguish between issues of pure traceability, and issues of assurance or integrity? Should traceability just include tracing data to its production source, or also tracing design requirements for data structures and preparation back to system-level requirements? Does traceability need to be two-way, giving the preparer visibility of the end use of data?

8.2. Development of Further Guidance for Safety Analysis of Data

This report has found that a body of guidance on analysis and management of data in safety-related systems does exist. However, this material appears to be disparate, spread over a number of different publications that often only touch on small areas of the topic. A number of the more useful sources have not yet been formally published. There appears to be some utility in drawing this guidance together, developing it and incorporating examples of best practice, to act as a reference work.

The overall aim of the document would be to provide guidance on how to construct and maintain an argument for the safety of systems that depend on data for their behaviour. Key areas to include in such a document would include:

- Techniques for identifying potentially hazardous faults and failure modes in data (perhaps based on the taxonomy presented in this report).
- Techniques, or adaptations of techniques, for identifying which potentially hazardous faults can manifest unsafe conditions at the system level.
- Strategies for avoiding the causes of such faults and managing the consequences.

The guidance could be based on satisfaction of the Def Stan 00-56 and Def (Aust) 5679 requirements [36, 10] and structured around a lifecycle process for safety management of data, similar to that proposed by Storey and Faulkner [47]. It could present a framework that would help identify which techniques were applicable at each stage of the system's product lifecycle. As data is only safety-related when considered in the context of a wider

system, it would also have to explain how data-focussed tools, techniques, processes and design strategies could fit in with other areas of engineering, management and safety assessment.

While this information would be useful as a stand-alone publication, it would also provide useful input to systems safety and systems engineering standards.

8.3. Application to Security

Many of the faults described in this document in a safety context could also be relevant in a security context, since unhandled faults that can occur accidentally could be exploited for malicious purposes by deliberate attacks. Examples include corrupting data, jamming or flooding a communications channel, or spoofing messages. It may be useful to examine whether the data fault taxonomy has any application in the security domain, or whether security-related work could inspire further refinements in the taxonomy.

8.4. Metadata for Safety Assurance

This report has noted the benefits of metadata: extra information added to a data set, to describe properties of that data. It would appear that there is significant potential for investigation of the types of metadata that would be useful to support safety assurance of data.

A key use of metadata is in conveying extra information about data that allows otherwise undetectable subtle value errors to be detected as inconsistency, formatting, or other types of fault. However, metadata could also be used to carry provenance information about the supply chain that produced the data, or the needs of the consumer of that data.

As an example, data could have metadata attached which declared the SIL level of the system that produced it. This is not especially useful as an indicator of correctness, as the data may be subsequently go stale or be modified (deliberately or otherwise), as it is transmitted, processed and stored prior to use. However, this metadata would be useful in letting subsequent links in the data handling chain identify how the data should be handled to preserve its original integrity, and in allowing the end user to decide how much trust should be placed in the information.

Metadata will be particularly useful for data in a system-of-systems context, where the component systems are not specifically designed to work together. In these cases a system architect cannot know in advance the provenance or integrity of a data supply chain, as might be the case in a conventionally designed system. Subtle faults in data could be a particular problem in message-passing systems-of-systems, where messages may be correctly formatted and come directly from an authorised source, but the earlier links of the data supply chain are obscured. The end user may have no independent way of validating the correctness of the data, i.e. whether a command they have been given will result in an unsafe outcome.

From the point of view of the majority of systems in a system-of-systems, much of the important data is third-party. It is not objective data that they interpret, or intent data that defines their behaviour, meaning that it is not likely to be captured by standard requirements analysis. Metadata provides a mechanism for third-party data to communicate requirements to the systems that process, communicate or store it, in terms of qualities such as those identified by DO-200A [3].

There would therefore be merit in work to develop an open framework for metadata to aid in the safety assessment of data at system run-time.

9. Conclusions

This project has reviewed published papers and standards to identify currently available guidance on the analysis of data in safety related systems. Themes from this source material have been drawn together and developed, leading to the following conclusions:

- The IEC 2382 definition of data as "a reinterpretable representation of information in a formalized manner suitable for communication, interpretation or processing," is a useful definition of what is meant by data, that can lead to an identification of the types of problem that can make data unfit for purpose, and potentially hazardous.
- Data may be considered in three categories, any of which may be safety-related:
    - Objective – describing objects in the internal or external environment of a system;
    - Intent – describing how a system is intended to behave; or
    - Third party – describing information used by other systems that is communicated or stored by the system in question, but not used or prepared by it.
- Introducing data to a system introduces hazards in that the system may behave exactly according to its specifications, with no fault or failure of hardware and software, but still behave in a way that is unsafe, due to faults in the particular data being used.
- Faults in data can be considered as conditions when the data available at time of use does not represent the true intent of the user. Faults may arise from a number of different causes and lead to a number of different outcomes, some of which may be hazardous at the system level.
- Faults may be considered either detectable or subtle. Subtle faults are those where the data is plausible, but wrong. While some faults may theoretically be detectable, the level of resources needed to achieve detection may not be practical.
- The emphasis when designing systems and analysing the safety of data should be to reduce the overall number of potential faults and the proportion of those faults that cannot be detected.
- A strategy for making subtle faults detectable is to build redundant information or extra knowledge into data elements, structures and formatting, such that subtle value faults may be detected as faults such as inconsistencies at a higher level of abstraction.
- If it is not possible to make subtle faults detectable, a system designer must rely on trust to ensure that data will be correct. An assurance argument can be built up about the process used to supply the data and ensure its correctness. This argument gives the provenance of the data.
- Metadata (data about data) may provide potential solutions, both to making more faults detectable and to supplying information about the provenance and handling requirements of safety-related data.

The literature review also found that there is a growing body of guidance concerning analysis and management of data in safety-related systems, but that this guidance is spread between many different documents dealing with different aspects of the problem. The project identified a need to consolidate and develop this material into a coherent set of best practice. It found several specific areas that could be addressed by future work, in particular the use of metadata and the application of safety concepts about data to the security domain (and visa versa).

The project also found a lack of published material to aid practitioners conducting hazard identification exercises for data. A taxonomy of data faults has been proposed to help fill this gap. The taxonomy is intended to be used as a checklist or set of guide words to help designers identify what faults could be present in their data. This would lead into an analysis of whether those faults would manifest at the system level in a hazardous manner, and if so, how they could be caused and prevented.

The taxonomy has been partially validated by use for classification of data-related factors in accident an incident reports. The exercise found that the taxonomy was sufficient to classify all identified data-related faults. However, faults were not identified in every category of the taxonomy and no validation has been provided that the taxonomy is optimal.

The taxonomy would benefit from further validation work, to check that it can represent all data faults encountered in real-life accidents. It also requires practical validation to show that it is a useful tool for practitioners.

10. Acronyms

AAIB	Air Accident Investigation Branch
AIS	Automatic Identification System
ARP	Aerospace Recommended Practices
ASSC	Formerly: Avionic Systems Standardisation Committee (scope now widened to include all real-time, high integrity software).
ATS	Air Traffic Services
BS	British Standard
CATZOC	Category of Zone of Confidence
CEE	Complex Electronic Element
CENELEC	European Committee for Electrotechnical Standardization
Def (Aust)	Australian Defence Standard
Def Stan	Defence Standard
DME	Distance Measuring Equipment
EN	European Norm
EPIRB	Emergency Position Indicating Radio Beacon
FLS	Field Loadable Software
GEIA	Government Electronics & Information Technology Association
GPS	Global Positioning System
GPWS	Ground Proximity Warning System
HGV	Heavy Goods Vehicle
IEC	International Electrotechnical Commission
IMA	Integrated Modular Avionics
ISO	International Standards Organisation
MAIB	Marine Accident Investigation Branch
Mil-Std	(US) Military Standard
MOD	Ministry of Defence
MRCC	Maritime Rescue Co-ordinating Centre
MV	Motor Vessel
OSGB36	Ordnance Survey Great Britain 1936
OTDR	On Train Data Recorder
RAIB	Rail Accident Investigation Branch
RTCA	Radio Technical Commission for Aeronautics
SHARD	Software Hazard Analysis and Resolution in Design
SOI	Special Operating Instruction
TOPS	Total Operations Processing System
VHF	Very High Frequency
WGS84	World Geodetic System of 1984
WON	Weekly Operating Notice
XML	Extensible Markup Language

11. Bibliography

[1] JSON website. http://www.json.org/ retrieved 4 September 2008.

[2] RTCA Special Committee 167. Software considerations in airborne systems and equipment certification. Recommendation DO-178B, RTCA, Inc, Washington, December 1992.

[3] RTCA Special Committee 181. Standards for processing aeronautical data. Recommendation DO-200A, RTCA, Inc, Washington, September 1998. Cited in Faulkner Storey (2003).

[4] RTCA Special Committee 181. Standards for aeronautical information. Recommendation DO-201A, RTCA, Inc, Washington, April 2000. Cited by Faulkner and Storey (2003).

[5] Boris Beizer and Otto Vinter. Bug taxonomy and statistics. Website, 2001. http://inet.uni2.dk/~vinter/bugtaxst.doc retrieved 2 Aug 08.

[6] Air Accident Investigation Branch. AAIB website, 2001-2008. http://www.aaib.gov.uk/sites/aaib/publications/formal_reports.cfm.

[7] Marine Accident Investigation Branch. MAIB website, 1998-2008. http://www.maib.gov.uk/publications/investigation_reports.cfm.

[8] Rail Accident Investigation Branch. RAIB website, 2006-2008. http://www.raib.gov.uk/publications/investigation_reports.cfm

[9] R.W. Butler and G.B. Finelli. The infeasibility of quantifying the reliability of life-critical real-time software. *IEEE Transactions on Software Engineering*, 19(1):3–12, Jan 1993.

[10] Tony Cant. Safety engineering for defence systems. Australian Defence Standard Def (Aust) 5679, Australian Commonwealth Department of Defence, Defence Science and Technology Organisation, Edinburgh, SA, Australia, March 2007. Issue 2, draft version 1.1 (issued for comment).

[11] R Chippendale. Air New Zealand Mcdonnell-Douglas DC10-30 ZK-NZP, Ross Island, Antarctica 28 November 1979. Aircraft Accident Report 79-139, Office of Air Accidents Investigation, New Zealand Ministry of Transport, Wellington, New Zealand, May 1980.

[12] International Electrotechnical Commission. Functional safety of electrical/electronic/programmable electronic safety-related systems - part 2: Requirements for electrical/electronic/programmable electronic safety-related systems. British Standard BS EN 61508-2:2002, British Standards Institution, March 2002.

[13] International Electrotechnical Commission. Functional safety of electrical/electronic/programmable electronic safety-related systems - part 3: Software requirements. British Standard BS EN 61508-3:2002, British Standards Institution, March 2002.

[14] International Electrotechnical Commission. Functional safety of electrical/electronic/programmable electronic safety-related systems - part 4: Definitions and abbreviations. British Standard BS EN 61508-4:2002, British Standards Institution, March 2002.

[15] International Electrotechnical Commission. Functional safety of electrical/electronic/programmable electronic safety-related systems - part 7 - overview of techniques and measures. British Standard BS EN 61508-7, British Standards Institution, March 2002.

[16] ITAA G48 Committee. GEIA-STD-0010: Standard best practices for system safety program development and execution. Committee Draft, June 2008.

[17] Alastair Faulkner. Safer data: the use of data in the context of a railway control system. In *Proceedings of the tenth Safety-critical Systems Symposium*, pages 217–230, 2002.

[18] Alastair Faulkner, P. A. Bennett, Ron Pierce, I. H. A. Johnston, and Neil Storey. The safety management of data driven safety related systems. In *Proc. 19th Int. Conf. Safecomp*, pages 86–95, Rotterdam, The Netherlands, October 2000.

[19] Alastair Faulkner and Ron Pierce. Is it data or is it software? In *Proceedings of the 19th International Safety System Conference*, pages 323–329, Huntsville, Alabama, September 2001. System Safety Society.

[20] Alastair Faulkner and Neil Storey. The role of data in safety-related railway control systems. In *Proceedings of the 19th International Safety System Conference*, pages 793–800, Huntsville, Alabama, September 2001. System Safety Society.

[21] Alastair Faulkner and Neil Storey. Data: An often-ignored component of safety-related systems. In *Proceedings of the MOD Equipment Assurance Symposium ESAS02*, Bristol, UK, October 2002. Ministry of Defence.

[22] International Organization for Standardization and International Electrotechnical Commission. Information technology - vocabulary - part 1: Fundamental terms. British Standard BS ISO/IEC 2382-1:1993, British Standards Institution, November 1993.

[23] Ken Frazer, Duncan Dowling, and Mike Ainsworth. Developing data management processes for safety critical systems. In *Proceedings of the 21st International Safety System Conference*, Ottawa, Canada, August 2003. System Safety Society.

[24] Dafydd Gibbon and Peter Ladkin. Comments on confusing conversation at Cali. Website, February. 1996. Retrieved 10 September 2008.

[25] Ian Glazebrook. Additional guidance and considerations on the application of RTCA DO-178B. Technical Report 2007-0419, ASSC, August 2007.

[26] CAA Safety Regulation Group. SW01 - regulatory objectives for software safety assurance in ATS equipment. In *CAP670 - Air Traffic Services Safety Requirements*. Civil Aviation Authority, June 2003.

[27] A. Harrison and R. H. Pierce. Data management safety requirements derivation. Technical report, Railtrack plc, June 2000. West Coast Route Modernisation Internal report. Cited by Faulkner (2001).

[28] Paul Hollow, John Mcdermid, and Mark Nicholson. Approaches to certification of reconfigurable IMA systems. In *10th International Symposium of the International Coucil on Systems Engineering*, Minneapolis, USA, July 2000.

[29] C M Holloway and C W Johnson. Why system safety professionals should read accident reports. In *Proceedings of the 1st International Conference on System Safety*, London, June 2006. Institution of Engineering and Technology.

[30] Peter Ladkin. News and comment on the Aeroperu B757 accident, AeroPeru Flight 603, 2 October 1996, November 1997. Website retrieved 5 September 2008. http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/ComAndRep/AeroPeru/aeroperu-news.htm.

[31] Bev Littlewood and Lorenzo Strigini. Validation of ultra-high dependability for software-based systems. *Communications of the ACM*, 36:69–80, Nov 1993.

[32] Maritime and Coastguard Agency. MCA website. http://www.mcga.gov.uk/c4mca/mcga07-home/shipsandcargoes/mcga-shipsregsandguidance/marinenotices.htm retrieved 19 June 2008.

[33] T Murray. Blueprint workshop report. Technical Report QINETIQ/S&E/AVC/CR031274, QinetiQ, May 2003.

[34] Ministry of Defence. Requirements for safety related software in defence equipment part 1: Guidance. Defence Standard Def Stan 00-55, Directorate of Standardization, Glasgow, August 1997. Issue 2.

[35] Ministry of Defence. Requirements for safety related software in defence equipment part 1: Requirements. Defence Standard Def Stan 00-55, Directorate of Standardization, Glasgow, August 1997. Issue 2.

[36] Ministry of Defence. Safety management requirements for defence systems - part 1: Requirements. Defence Standard Def Stan 00-56, Directorate of Standardization, Glasgow, June 2007. Issue 4.

[37] Ministry of Defence. Safety management requirements for defence systems - part 2: Guidance on a means of complying with part 1. Defence Standard Def Stan 00-56, Directorate of Standardization, Glasgow, June 2007. Issue 4.

[38] Ministry of Defence. Guidance on the assurance of safety in systems containing complex electronic elements in support of Def Stan 00-56 Issue 4. Final draft for approval, July 2008.

[39] Department of Defense. Standard practice for system safety. Military Standard MIL-STD-882D, United States Department of Defense, February 2000.

[40] Aeronautica Civil of the Republic of Colombia. Controlled flight into terrain American Airlines Flight 965 Boeing 757-223, N651AA, near Cali, Colombia, December 20, 1995. Aircraft accident report, Published on the web by Peter Ladkin, Santa Fé de Bogota, Colombia, September 1996. Retrieved from http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/ComAndRep/Cali/calirep.html on 9 July 2008.

[41] David Pumfrey. *The Principled Design of Computer System Safety Analyses*. PhD thesis, Department of Computer Science, University of York, Sept 1999.

[42] Felix Redmill. History and legacy of IEC 61508. *Safety Systems*, 17(2):37–41, January 2008.

[43] Carolyn Salmon and Clive Lee. The certification of systems containing software developed using RTCA DO-178B. Technical Report ASSC/12/0013, ASSC, June 2006.

[44] Roger C Short. Safety assurance of configuration data for railway signal interlockings. In *Proceedings of the 1st International Conference on System Safety*, London, June 2006. Institution of Engineering and Technology.

[45] A J Simpson and J Stoker. Safety challenges in flying UAVs (unmanned air vehicles) in non segregated airspace. In *Proceedings of the 1st International Conference on System Safety*, London, June 2006. Institution of Engineering and Technology.

[46] Neil Storey. Data-driven systems - the state of the ark? *Safety Systems*, 17(2):28–31, January 2008.

[47] Neil Storey and Alastair Faulkner. Data management in data-driven safety-related systems. In *Proceedings of the 20th International Safety System Conference*, pages 466–475, Denver, Colorado, August 2002. System Safety Society.

[48] Neil Storey and Alastair Faulkner. Characteristics of data in data-intensive safety-related systems. In *Proceedings of the 22nd International Conference SafeComp 2003*, pages 396–409, Edinburgh, September 2003.

[49] Nassim Nicholas Taleb. Fooled by randomness – the hidden role of chance in life and in the markets. Cited on Wikiquote, 2001. http://en.wikiquote.org/wiki/Nassim_Nicholas_Taleb.

[50] Andrew S. Tanenbaum. Computer networks. Cited on Wikiquote, 1981. http://en.wikiquote.org/wiki/Andrew_S._Tanenbaum.

[51] Mark Templeton. Safety integrity of data. Master's thesis, Department of Computer Science, University of York, York, UK, September 2007.

[52] John Tillotson. System safety and management information systems. In Felix Redmill and T. Anderson, editors, *Aspects of Safety Management: Proceedings of the Ninth Safety-Critical Systems Symposium, Bristol,*

UK, 6-8 February 2001, pages 13–34, Secaucus, NJ, USA, 2001. Safety Critical Systems Club, Springer-Verlag New York, Inc.

[53] D. Welbourne and N. P. Bester. Data for software systems important to safety. *GEC Journal of Research*, 12(1):50–57, 1995. Cited in Faulkner & Storey (2001).

Appendix A. Taxonomy Coverage Chart

The coverage chart below shows how fault categories from the source material [5, 27, 20, 17, 41, 12, 3, 38] have been reflected in the new data fault taxonomy and where new categories have been added to fill in perceived gaps in the taxonomy. Rows/columns with no coverage are labelled in *italics*.

		1.0.0 *Meaning*	1.1.0 Value	1.1.1 *Meaningless*	1.1.2 Inaccurate data value	1.1.3 Association error	1.2.0 Insufficient data resolution	1.2.1 *Aliasing*	1.3.0 *Ambiguity*	1.3.1 *Multiple interpretation*	1.3.2 *Violation of uniqueness*	1.3.3 *Lack of precision*	1.4.0 *Inconsistency*	1.4.1 Inconsistent between instances	1.4.2 Inconsistent over series	1.4.3 Inconsistent between diverse data	1.5.0 Omission	1.5.1 Incomplete data	1.6.0 Commission	1.6.1 Repetitions	1.6.2 Data overrun	2.0.0 Format	2.1.0 Wrong scaling or units	2.2.0 *Wrong datum*	2.3.0 Wrong type	2.4.0 Data out of range	2.5.0 *Wrong Language*	2.6.0 *Wrong grammar*	2.7.0 Data out of sequence	3.0.0 Timing	3.1.0 Omission	3.1.1 Access rights violation	3.1.2 Existence	3.1.3 Availability	3.1.4 Loss of data	3.2.0 Commission	3.2.1 Repeated receipt of data	3.3.0 Early	3.4.0 Late	3.4.1 Data beyond its period of validity.	3.5.0 Data out of sequence	4.0.0 Provenance	4.1.0 Masquerade	4.2.0 Unknown provenance
Software Bugs (Beizer)	Wrong type																								X																		
	Type transformation																																										
	Scaling, units																						X																				
	Constant value		X																																							
	No such resource					X																											X										
	Wrong resource type					X																																					
	Wrong object accessed					X																																					
	Access rights violation																														X												
	Data-flow anomaly																																							X			
	Interlock bug																																										
	Object boundary or structure																	X			X																						
Railway Infrastructure	Omission																X																										
	Spurious data																		X	X																							
	Positioning errors		X																																							
	Topological errors															X																											
	Addressing errors					X																																					

IEC 61508			SHARD					Status Data						External Data									
Deletion	Repetitions	*Transmission errors*	Value	Late	Early	Commission	Omission	Volume fault	Timing fault	*Propagation fault*	Combinational fault	Sequential fault	Mode fault	Timely	Inconsistent	Availability	Reference error	Existence	Value errors	Labelling errors	Type errors		
																						1.0.0	*Meaning*
			X																X		X	1.1.0	Value
																						1.1.1	*Meaningless*
																						1.1.2	Inaccurate data value
																	X			X		1.1.3	Association error
																						1.2.0	Insufficient data resolution
																						1.2.1	*Aliasing*
																						1.3.0	*Ambiguity*
																						1.3.1	*Multiple interpretation*
																						1.3.2	*Violation of uniqueness*
																						1.3.3	*Lack of precision*
																						1.4.0	*Inconsistency*
																						1.4.1	Inconsistent between instances
															X							1.4.2	Inconsistent over series
													X									1.4.3	Inconsistent between diverse data
X							X															1.5.0	Omission
											X											1.5.1	Incomplete data
						X																1.6.0	Commission
	X																					1.6.1	Repetitions
																						1.6.2	Data overrun
																						2.0.0	Format
																						2.1.0	Wrong scaling or units
																						2.2.0	*Wrong datum*
																						2.3.0	Wrong type
																						2.4.0	Data out of range
																						2.5.0	*Wrong Language*
																						2.6.0	*Wrong grammar*
				X	X							X										2.7.0	Data out of sequence
																						3.0.0	Timing
							X		X													3.1.0	Omission
																						3.1.1	Access rights violation
																		X				3.1.2	Existence
																X						3.1.3	Availability
X																						3.1.4	Loss of data
						X																3.2.0	Commission
	X																					3.2.1	Repeated receipt of data
					X			X														3.3.0	Early
				X				X						X								3.4.0	Late
																						3.4.1	Data beyond its period of validity.
												X										3.5.0	Data out of sequence
																						4.0.0	Provenance
																						4.1.0	Masquerade
																						4.2.0	Unknown provenance

MOD CEE Guidance					DO-200A																
Invalid relationships between different data items	Untimely data	Incorrect relationships between successive data items	Range errors	Type errors	Incorrect format	Incomplete data	Untimely data	Unknown provenance	*Data altered after preparation*	Insufficient data resolution	Inaccurate data value	Masquerade	Delay	*Corruption*	Resequencing	Insertion		
																	1.0.0	*Meaning*
																	1.1.0	Value
																	1.1.1	*Meaningless*
											X						1.1.2	Inaccurate data value
																	1.1.3	Association error
										X							1.2.0	Insufficient data resolution
																	1.2.1	*Aliasing*
																	1.3.0	*Ambiguity*
																	1.3.1	*Multiple interpretation*
																	1.3.2	*Violation of uniqueness*
																	1.3.3	*Lack of precision*
																	1.4.0	*Inconsistency*
																	1.4.1	Inconsistent between instances
		X															1.4.2	Inconsistent over series
X																	1.4.3	Inconsistent between diverse data
																	1.5.0	Omission
						X											1.5.1	Incomplete data
																X	1.6.0	Commission
																	1.6.1	Repetitions
																	1.6.2	Data overrun
					X												2.0.0	Format
																	2.1.0	Wrong scaling or units
																	2.2.0	*Wrong datum*
				X													2.3.0	Wrong type
			X														2.4.0	Data out of range
																	2.5.0	*Wrong Language*
																	2.6.0	*Wrong grammar*
															X		2.7.0	Data out of sequence
	X						X										3.0.0	Timing
																	3.1.0	Omission
																	3.1.1	Access rights violation
																	3.1.2	Existence
																	3.1.3	Availability
																	3.1.4	Loss of data
																X	3.2.0	Commission
																	3.2.1	Repeated receipt of data
																	3.3.0	Early
													X				3.4.0	Late
							X										3.4.1	Data beyond its period of validity.
															X		3.5.0	Data out of sequence
																	4.0.0	Provenance
												X					4.1.0	Masquerade
								X									4.2.0	Unknown provenance

							1.0.0 *Meaning*
							1.1.0 Value
							1.1.1 *Meaningless*
							1.1.2 Inaccurate data value
							1.1.3 Association error
							1.2.0 Insufficient data resolution
							1.2.1 *Aliasing*
							1.3.0 *Ambiguity*
							1.3.1 *Multiple interpretation*
							1.3.2 *Violation of uniqueness*
							1.3.3 *Lack of precision*
							1.4.0 *Inconsistency*
							1.4.1 Inconsistent between instances
							1.4.2 Inconsistent over series
						X	1.4.3 Inconsistent between diverse data
							1.5.0 Omission
							1.5.1 Incomplete data
							1.6.0 Commission
							1.6.1 Repetitions
							1.6.2 Data overrun
							2.0.0 Format
							2.1.0 Wrong scaling or units
							2.2.0 *Wrong datum*
							2.3.0 Wrong type
							2.4.0 Data out of range
							2.5.0 *Wrong Language*
							2.6.0 *Wrong grammar*
					X		2.7.0 Data out of sequence
							3.0.0 Timing
							3.1.0 Omission
							3.1.1 Access rights violation
							3.1.2 Existence
				X			3.1.3 Availability
				X			3.1.4 Loss of data
							3.2.0 Commission
					X		3.2.1 Repeated receipt of data
							3.3.0 Early
					X		3.4.0 Late
							3.4.1 Data beyond its period of validity.
					X		3.5.0 Data out of sequence
							4.0.0 Provenance
							4.1.0 Masquerade
							4.2.0 Unknown provenance