Safety Systems, vol. 21, no. 1, pp. 4-7, September 2011

Safe Data: Recognising the Issue

Author: James Inge

Over the years, several commentators have raised concerns about unsafe data (including Neil Storey in this publication [7]). They pointed out that despite the potentially harmful effects of bad data, the issue was covered poorly in mainstream standards, with little guidance available about how to manage the risks that could arise from them. Systems can use data to describe the environment that they interact with, or their own physical configuration. Data can also be used to direct how systems behave, in a manner similar to software. When data determines some of the behaviour of a system, that data may affect safety at the system level. Data can be bad for a number of reasons. It may be generated incorrectly, with bad values or formatting; it may be corrupted in transmission or storage, or it may become stale through changes in the real-world values it represents. Just as design flaws in software can make a system behave unsafely without any hardware fault, bad data can cause unsafe behaviour without any fault in either the hardware or software. And, as with software, the overall system can often be too complicated for safety to be shown through testing alone.

Is this a real issue? Aside from the humorous anecdotes about drivers being led astray by their satnav systems, errors in in-car navigation databases have led to fatal consequences [2]. Similarly, problems with air navigation data have been a factor in several passenger aircraft crashes, such as that of American Airways Flight 965 near Cali in Colombia [1]. The self-destruction of Ariane 5 Flight 501 involved diagnostic data being interpreted as flight data [5]. Just as software modules are sometimes reused in contexts that their designers had not considered (another factor in the Ariane 5 incident), data may be re-purposed. The new applications may not have been anticipated by the designers of the data set, and may place more trust in the data than is merited. UK post codes give sufficient precision for dispatch of letters to a street, but was the data set intended to support dispatch of emergency services in life-threatening circumstances?

Data considerations were found in around 10% of the incidents analysed in a review of 192 reports published by the UK's Air, Rail and Marine Accident Investigation Branches up to August 2008. These considerations were a combination of causal factors and other safety-relevant factors that were not causal but were commented on in the investigation report, such as problems with emergency response. As well as equipment being configured with incorrect values, they included data that was missing, had not been updated to match real-world changes, referred to the wrong object, or came from an invalid source. Many of the incidents involved automated data processing, but others involved human processing of data [4].

The challenges data brings to a safety-related system are similar to those brought by software, but at a greater degree of abstraction. Even if the solutions are still elusive, at least the problems associated with software-driven functionality are well recognised:

- Unlike traditional hardware-based systems, software-based systems do not exhibit continuous behaviour, so testing with one set of inputs may not represent other circumstances that differ even slightly.
- Software allows functionality to be arbitrarily complex, making it impractical to test all possible inputs for many practical systems.
- As only the hardware components of systems can directly pose a threat to physical safety, software is sometimes seen as being somewhat detached from the hazards posed by a system, even when it is the software that determines the behaviour of the hardware.
- The software and hardware components of a system are often designed separately from each other, by personnel with different skills and knowledge. This can make it difficult for individual designers to predict the system-level implications of their decisions.

This last point is important. Software and hardware work together and safety is an emergent property of the system as a whole, rather than of its components. If software is developed in isolation from the hardware, and without regard for the context in which the system will be used, it can be hard to predict what impact the software will have on the safety of the system. The same applies to data. It would be nice if such data could be prepared, analysed and tested or proven correct at design-time with the rest of the system, to check that it would lead to safe behaviour. In practice this is often impractical, or even impossible. The data used by a system will often be generated long after that system is deployed. It may be prepared by end-users or other third parties who are independent of the original system design effort, and have little understanding of what impact nuances in the data might have on the system's safety (or other attributes, such as reliability). The data may not even be understood to form part of the system.

However, once data is recognised as forming part of a system, with an impact on its safety, the system can be designed to reduce the likelihood of bad data leading to unsafe behaviour. Techniques such as checksums or cyclic redundancy checks can detect unintended changes to data. Formatting, data structures and metadata can be designed to allow for consistency checking. Refresh rates can be set so that data keeps up-to-date with the real-world quantities it is supposed to represent. Management arrangements can be set up to ensure that data comes only from trustworthy sources.

In the past, few safety standards have given practical guidance on dealing with data, with RTCA standard DO-200A *Standards for Processing Aeronautical Data* being an exception. DO-200A describes safety-relevant attributes of data such as accuracy, resolution, assurance level, traceability, timeliness, completeness and format. It also provides a framework for describing the supply chain for data, including how it is generated, processed, passed through intermediaries, and eventually delivered to the end user [6]. More recently, requirements related to data handling have been included throughout the 2010 edition of IEC 61508-3. It also includes a new Annex G, focussing on configuration data, which gives guidance for tailoring lifecycles associated with data-driven systems [3].

Before any of these techniques can usefully be put into effect, one must have an understanding of what could go wrong with the data: what problems it could cause that might lead to hazardous behaviour at the system level. A taxonomy of data faults is proposed to aid in safety analysis (Table 1). The taxonomy is structured as a hierarchy, to allow increasingly detailed levels of application. It is intended to be used as a series of prompts or guidewords, to help a designer understand what might go wrong with the data used by their system. This information can feed bottom-up safety analyses such as

HAZOPS or FMECA, to investigate whether data faults could lead to adverse system-level behaviour, and to reveal areas where precautions need to be taken [4].

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. conflicting sources of same information
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 additionally 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 1. Data fault taxonomy.

Data can have an impact on safety at the system level, if systems use data to determine their behaviour. While this could apply to any type of system, as systems become more complex, automated and data-driven, it is likely that the impact of data on safety will increase. Making such systems safe requires both a recognition of the role of data as an element of the system and an understanding of how that data could go wrong. With this in place, designers and systems engineers can ensure that overall systems take account of the impact of potential problems with the data they rely on.

References:

[1] 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, September 1996. Retrieved 9 July 2008 from http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/ComAndRep/Cali/calirep.html.

[2] BBC. Girl, 4, died in driver sat-nav error crash in Blackrod. BBC News website, February 2011. Retrieved 21 May 2011 from http://www.bbc.co.uk/news/uk-england-12360687.

[3] IEC. Functional safety of electrical/electronic/programmable electronic safety-related systems - part 3: Software requirements. British Standard BS EN 61508-3:2010, British Standards Institution, May 2010.

[4] INGE, J. Improving the analysis of data in safety-related systems. Postgraduate diploma report, University of York Department of Computer Science, York, UK, September 2008. http://safety.inge.org.uk/publications.htm.

[5] LIONS, J. L. Ariane 5 Flight 501 Failure. Report by the inquiry board, European Space Agency, Paris, July 1996.

[6] RTCA SPECIAL COMMITTEE 181. Standards for processing aeronautical data. Recommendation DO-200A, RTCA, Inc, Washington DC, USA, September 1998.

[7] STOREY, N. Data-driven systems - the state of the ark? *Safety Systems, vol. 17*, no. 2, pp. 28–31, January 2008.

Biography:

James Inge is head of the Ship Safety Management Office at the UK MOD. He has a background in acquisition safety policy, including development of JSP 430 and DefStan 00-56. He studied Safety Critical Systems Engineering at the University of York and is a member of the IET/BCS/SaRS Independent Safety Assurance Working Group.