Putting Trust into Safety Arguments

Jane Fenn and Brian Jepson
BAE Systems, Warton Aerodrome,
Preston, Lancashire, PR4 1AX

Abstract

This paper describes one development of a concept that emerged from the Defence and Aerospace Research Partnership to enhance safety arguments by identifying and managing the argument's dependence on safety evidence.

1 Introduction

The application of Safety Engineering methodologies is relatively immature as an engineering discipline. Techniques are inconsistently applied within and across domains. Reaching consensus on 'best practice' is difficult, if not impossible, consequently, a plethora of standards have emerged which reflect various 'flavours' of safety engineering. Typically, these present a 'cook book' approach to safety techniques; leaving the rationale for selection and appropriate combinations of techniques largely implicit. Similarly, there is an implicit, and, as yet, unproven assumption in these standards that a good process necessarily yields a safe product. Many critics of these standards have observed that, whilst a good process is necessary, it is not sufficient to ensure a safe product.

A characterisation of these standards can be used to draw out some of the contentious areas and plot a timeline of the development of safety standards as we, in the UK defence industry, brace ourselves for migrating to issue 3 of Defence Standard 00-56[1].

Perhaps it's best to start with a standard that has been quite widely used within the safety community: DEF STAN 00-56 issue 2[2]. This standard proposes managing safety by means of risk matrices based on failure rates and hazard consequences and provides guidance on determining tolerability criteria. Systematic errors are handled by way of Safety Integrity Levels; SIL 4 representing the highest safety risk systems and SIL 1 the lowest. These SILs are mapped across to development processes, of which the software aspects are addressed in more detail in DEF STAN 00-55[3]. The explicit record of the safety of the system, as declared by the developer, is contained in the safety case.

The Australian equivalent standard is DEF AUST 5679[4]. A similar schema is used where 'Levels of Trust' are mapped to SILs for software; SIL 6 presenting the highest risk and SIL 1 lowest. Examples are then given for design, implementation and verification techniques, based on allocated SIL. Whilst this standard doesn't have an explicit safety case, the 'Safety Analysis Plan' requires some of the same types of information.

The more generic European standard IEC 61508[5] has SILs too, though these are somewhat different, with SIL4 systems intended to have the lowest hazardous failure rate. Detailed techniques are then presented, with recommendation categories dependent on SIL, which are intended to address both random and systematic failures.

Our American colleagues use MIL-STD-882. Issue C [6] of the standard identified 'Software Control Categories' which are similar to SILs, but then gives no advice on how these should be handled, other than referring out to DO-178B[7]. Issue D [8] of the standard simply requires the developer to identify, evaluate and reduce 'mishap' risk. The standard contains a 'Safety Compliance Assessment' which fulfils some of the role of a safety case.

Whilst it is not a safety standard in itself, it is increasingly common for companies to offer compliance with DO-178B in response to safety requirements. This standard uses 'Development Assurance Levels' (DAL) to reflect the rigour of process required, with Level A being the highest and Level E lowest. Tables of techniques are provided in the annexes of the standard, indicating which should be used for each DAL, and noting independence requirements for these techniques at higher levels. The 'Software Accomplishment Summary' contains some of the same elements as a safety case.

None of the above standards offers a rationale for these choices of techniques. Various authors have criticised the approach, including Penny and Eaton[9], who said *"...there is a need to shift from documenting how hard people have tried to develop a system to providing evidence and argument about the behaviour of that system"*.

Support for this argument certainly seems to be growing, as demonstrated by the change in emphasis of some of the later standards such as the CAA's SW01[10]. This standard has a concept of 'Assurance Evidence Levels' (AEL) which "identify the depth and strength of evidence that must be made available from the software lifecycle for the equipment approval process". Dependent on AEL, an indication on the type of 'direct' testing and analysis necessary is provided, as well as the requirement for 'indirect' evidence of the process coverage. Whilst this standard clearly makes product evidence its mainstay, it still does not deal with the rationale behind the choices or composition of evidence to support a safety argument.