# *HazLog*: Tool support for hazard management

**Christian Hamoy**[⋆], **David Hemer**[†] **and Peter Lindsay**[⋆]

[⋆]School of Information Technology and Electrical Engineering
The University of Queensland.
Brisbane. Queensland 4072. AUSTRALIA.
Email: `Peter.Lindsay@itee.uq.edu.au`
[†]School of Computer Science
The University of Adelaide, SA, Australia, 5005
Email: `hemer@cs.adelaide.edu.au`

## Abstract

Industry is increasingly adopting software and system safety standards that mandate the use of hazard logs in the development and operation of safety critical systems. Hazard logs are used to record and track the results of hazard analysis and risk assessment throughout the lifecycle of the system. Even relatively simple systems give rise to large amounts of information and the need for tools to support the management of hazard logs.

Def(Aust) 5679 is the Australian Department of Defence's standard for procurement of computer-based safety critical systems. Def(Aust) 5679 has particular requirements for the nature of the information that needs to be tracked in the hazard log, and on the way that levels of trust (LOT) and safety integrity levels (SIL) are assigned to safety requirements.

This paper describes a prototype tool, called *HazLog*, that has been built on top of the DOORS tool in order to support the Def(Aust) 5679 hazard management process. DOORS is a requirements management tool which is already widely used within the Australian Department of Defence. Because many users will already be familiar with DOORS, learning to use the *HazLog* tool is expected to be easy. The tool helps users structure their hazard logs in the form required by Def(Aust) 5679, and helps them check the LOT and SIL rules from Def(Aust) 5679.

## 1 Introduction

### 1.1 Motivation

To gain a suitable level of assurance that a system is safe, a rigorous approach to identifying hazards and controlling the effects of these hazards must be followed. A number of industry standards (MIL-STD 882C 1993, MOD 00-56 1996, DEF(Aust) 5679 1998, IEC 61508 1998) define rigorous processes that aim at identifying and controlling hazards. An important element in all of these standards is documenting and tracking hazards and their associated resolutions. We will refer to this part of the overall process as *hazard management*.

Hazard management is the process of identifying, recording, analysing and subsequently implementing measures to control the effects of hazardous situations in systems where the lives of people are at risk (Leveson 1995). Hazard management is an instrumental part of the overall process of ensuring the safety of a system. Hazard management should occur across the entire system life-cycle, starting from requirements analysis, through to acquisition, in-service support and disposal.

David Hemer was previously in the School of Information Technology and Electrical Engineering, The University of Queensland.

While hazard management is pivotal in the overall process, it does typically require a lot of paper work to record and track the information. Since hazard management occurs at various levels and across the entire system, it is important to track the relationships and associations between the various sources of information and how they are linked with one another. For large, complicated systems, with many hazards, recording, analysing and tracking all of these hazards is a very time consuming and tedious process. Furthermore, because of the large volumes of information and complex interrelationships between this information, the hazard management process is prone to human error.

For these reasons tool support for hazard management is highly desirable in order to cut the time and costs of working through the hazard management process, and to reduce the possibility of human error. A software tool would help to increase efficiency as well as making it easier for users to keep track of the many document inter-relationships, which are required by the hazard management process.

### 1.2 Existing tool support

Currently there is no software tool available that fully supports the specific hazard management requirements of the Australian Defence Standard (DEF(Aust) 5679 1998). The closest candidate is HVR's *Cassandra* Hazard Management System (HVR 2000), which is designed to support the UK Ministry of Defence's standard (MOD 00-56 1996). Cassandra's features may be utilised to a certain degree but fall short of true Def(Aust) 5679 support.

A report was compiled by the Software Verification Research Centre (SVRC) (SVRC Services 2001), that provided an in-depth look, by way of a case study, at Cassandra, its features, and its suitability for use in conjunction with Def(Aust) 5679. The paper concluded that although Cassandra did provide many useful hazard management features, there were a number of deficiencies when attempting to support the process specified in Def(Aust) 5679.

In particular, while Cassandra has a customisable risk classification table, which can tailored to accomodate a variety of different hazard analysis approaches, it is not flexible enough to accomodate the two levels of risk indicators used in Def(Aust) 5679 (see Section 2 for more details). A more detailed discussion of Cassandra and its shortcomings with respect to Def(Aust) 5679 are given in Section 3.

In light of the identified shortcomings of Cassandra, our goal was to design and implement a software tool that offers full support for the Def(Aust) 5679 hazard management process. Enlightened by the aforementioned report on Cassandra, the first stage was to convert the shortcomings of Cassandra (with respect to support for Def(Aust) 5679) into a list of requirements for our hazard management software tool, which would provide full support for Def(Aust) 5679.

### 1.3 Overview

In this paper we describe *HazLog*, a prototype hazard management tool. The conceptual design for *HazLog* was driven by the needs of the Def(Aust) 5679 process, and was also influenced by the shortcomings of existing tools, in particular Cassandra. *HazLog* has been implemented as a collection of DOORS modules, with extra code for performing checks on the inputted hazard data. DOORS (Telelogic DOORS 2004) was chosen as the basis for the *HazLog* for two main reasons. Firstly it is already being used within the Australian Defence industry. Secondly it provides support for traceability analysis, object linking and multiple views, which are all useful for hazard management. We illustrate the key design features of *HazLog* with a simple example.

In Section 2 we give a brief overview of the relevant concepts of the Def(Aust) 5679 standard. In Section 3 we give an overview of Cassandra, listing its shortcomings with respect to Def(Aust) 5679. In Section 4 we introduce an example, concerning a storage facility for fuel and the associated hazards. This example is used later to illustrate the key parts of the tool. In Section 5 we describe the design of the *HazLog* tool, including screendumps showing the tool is use. In Section 6 we explain some of the checks that are performed automatically by the tool, used to ensure the integrity of the data entered by the user, and to check that safety integrity level assignments are reasonable.

## 2 Def(Aust) 5679

### 2.1 Overview

Def(Aust) 5679 (DEF(Aust) 5679 1998) is an Australian Defence standard used in the development of safety critical computer-based systems. Released in 1998 it borrows many concepts from other standards, in particular the UK Ministry of Defence standards (MOD 00-55 1997, MOD 00-56 1996). However Def(Aust) 5679 makes a fundamental departure from the standards on which it is based, in that it differentiates between two level of hazards (system-level and component-level), with different kinds of risk indicators assigned at the two levels (levels of trust and safety integrity levels respectively). This is quite different to the Ministry of Defence standards (and other related standards) which do not make this distinction between different levels of hazards, and only have one kind of risk indicator (safety integrity level). Again this is one of the main reasons why Cassandra is not suitable, since it is based on the model of a single risk indicator. For a detailed comparison of safety standards see (Wabenhorst & Atchison 1999).

Def(Aust) 5679 describes a process for analysing, designing, implementing and maintaining safety critical computer-based systems. We focus on the activities relating to developing and maintaining a safety case, in particular the hazard analysis and risk assessment tasks (DEF(Aust) 5679 1998, Sections 12-15). The standard describes the following three main hazard identification and risk assessment tasks:

- preliminary hazard analysis
- system hazard analysis
- system integrity assessment

In the remainder of this section we give a brief overview of each of these task, highlighting the information that needs to be recorded for each task.

### 2.2 Preliminary hazard analysis

The aim of the preliminary hazard analysis is to identify the possible *accidents* that can occur within the system, and the *system-level hazards* that can lead to these accidents. For each accident, an associated severity is determined. Def(AUST) 5679 distinguishes between the following four severity levels:

- Catastrophic;
- Fatal;
- Severe;
- Minor

Each accident has an associated *accident sequence*, which is a chain of events, such as system-level hazards, together with events external to the system, which can lead to an accident. The external events, referred to as *external co-effectors*, can sometimes be given an associated probability. Such probabilities can be combined to determine the likelihood that a system hazard will lead to an accident.

System Hazards are states of the system that can lead to an accident. For each system hazard there is an associated *system safety requirement*, which is a "negation" of the hazard stating conditions that must be met to ensure the particular hazardous state does not occur.

Each system safety requirement (SSR) is assigned a *level of trust* (LOT), which represents the desired level of confidence to be provided that the system meets the safety requirement. The LOT for an SSR is calculated by considering the accident sequences associated with the corresponding hazard. The LOT is a function of the severity of the resulting accident and the chance that the accident will occur, given the occurence of the associated system hazard (see Table 1). Def(Aust) 5679 defines seven levels of trust, labelled $T_0, T_1, .., T_6$. The lowest level ($T_0$) represents the situation where the system function has no safety critical implications (i.e. it cannot result in any System Hazards). The other levels range in the required level of trust from $T_1$ (the lowest) to $T_6$ (the highest).

A default LOT is assigned, based on the highest LOT of the accident sequences associated with the SSR. This may be reduced by considering the likelihood of accidents occurring, based on the probabilities that the external co-effectors leading to the accident occur.

### 2.3 System hazard analysis

The aim of the system hazard analysis stage is to decompose system level hazards into a number of *component level hazards.* Each system level hazard may be associated with multiple component hazards, and each component hazard may be associated with multiple system hazards.

While not being overly prescriptive, Def(Aust) 5679 does note that fault tree analysis (as well as other techniques such as cause-consequence analysis) is suitable for decomposing system hazards.

| | DEFAULT LEVEL | ACCIDENT PROBABILITY | | |
|---|---|---|---|---|
| ACCIDENT SEVERITY | | $> 10^{-2}$ | $10^{-2} - 10^{-4}$ | $< 10^{-4}$ |
| Catastrophic | $T_6$ | $T_6$ | $T_5$ | $T_4$ |
| Fatal | $T_5$ | $T_5$ | $T_4$ | $T_3$ |
| Severe | $T_4$ | $T_4$ | $T_3$ | $T_2$ |
| Minor | $T_3$ | $T_3$ | $T_2$ | $T_1$ |

Table 1: Levels of Trust Assignment

Each component-level hazard also has an associated *component safety requirement* (CSR), which capture safety requirements at the component level. Like SSRs, a CSR is a negation of a component hazard, representing a condition that must be met to ensure the hazard does not occur.

Def(Aust) 5679 notes that CSRs may also be derived from other sources, such as safety requirements specified by the user, or those in design standards. However it is not clear how such CSRs are managed within the framework of Def(Aust) 5679. For this version of the tool design we make the simplifying assumption that each CSR has a corresponding component hazard. We discuss this issue further in Section 7.

### 2.4 Safety integrity assessment

The aim of the safety integrity assessment stage is to allocate a *safety integrity level* (SIL) to each CSR, and more importantly to justify the allocation. A SIL gives a qualitative indication of the level of rigour that was (or will be) applied to provide assurance that the CSR is met. Unlike LOTs, which we recall are calculated automatically, SILs are assigned by the system developer. However this allocation must be justified, and it must be consistent with the overall level of assurance required for the system.

Corresponding to the seven LOTs, there are seven SILs, labelled $S_0, S_1, .., S_6$, where $S_6$ is the most demanding and $S_0$ is the least demanding. A SIL gives an indication of the kinds of engineering techniques that must be applied in developing and assuring the correctness etc. of the corresponding component. For example, for a $S_6$ software component fully formal design verification would be required, whereas for a $S_0$ component, no specific methods beyond the usual sound engineering principles are required.

To ensure that the SIL allocation is reasonable, we must perform a cross check against LOT of the SSR from which the CSR was derived. By default the SIL should be the same as the LOT of the associated SSR. However, under certain circumstance, and with proper justification, it is reasonable to assign a SIL lower that the corresponding LOT. The standard lists a number of rules that must be satisfied when allocation SILs (DEF(Aust) 5679 1998, Section 15). The tool implements a subset of the SIL allocation checks (see Section 6 for more details).

## 3 Cassandra

The Cassandra hazard management system (HVR 2000) is a software tool used to identify, assess and analyse hazards and accidents, as well as subsequent risk management procedures throughout the equipment life cycle. Cassandra was originally designed for military purposes, supporting UK Ministry of Defence Standard 00-56 (MOD 00-56 1996) and US Military Standard 882C (MIL-STD 882C 1993). Cassandra has since been adopted by other industries such as Aerospace, Rail, Oil and Gas.

Cassandra is appropriate for high-level hazard data management and for use as a hazard, accident and control data repository. Some of the useful features of the tool, as noted in (SVRC Services 2001), include:

1. Independent recording of hazards and accidents at any level of system design;
2. Recording of hazard and accident control actions;
3. Support for assessment of hazard and accident risk before and after control actions are in place, based on MIL-STD 882C or UK DEF STAN 00-56;
4. A user definable risk assessment approach that allows tailoring of probability, severity categories, risk classes and the risk matrix;
5. Report generation with various report filters to create reports on different aspects of the project database;
6. Exporting facilities of project data. This allows data to be shared in a common format and integrated with external tools such as a requirements database;
7. Merging facilities of project files, permitting merging of different project databases;
8. Management of a reference database for hazards, accidents and controls.

While Cassandra provides very good tool support for hazard management for a number of safety standards, a study conducted by the SVRC (SVRC Services 2001) identified a number of shortcomings that make it unsuitable for application to the Def(Aust) 5679 process. These identified shortcomings include:

1. Cassandra allows a hazard to be linked to an accident, but this is not sufficient for representing accident sequences;
2. The inability to record relationships between hazards. This is important where causal relationships between the hazards exist at different levels of system design such as system hazards that are decomposed into several component hazards;
3. The risk classification matrix (how risk control levels are calculated) provided by Cassandra would need considerable modifications to accommodate the Def(Aust) 5679 LOT/SIL dichotomy;
4. The limited support for recording safety requirements;

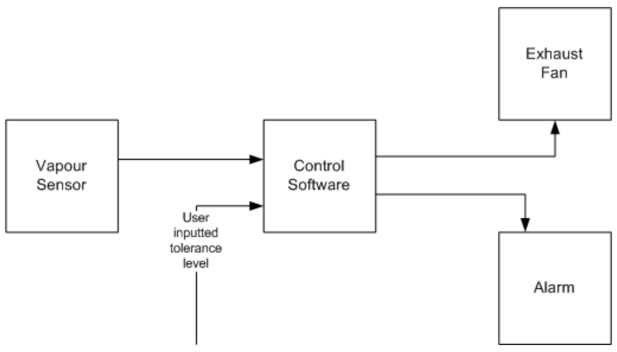


Figure 1: Data flow model for fuel store room

5. The limited search facilities through the hazard, accident and control database. A database search is only possible through the report generation facility of the tool.

The main goal of our project was to build a hazard management software tool that supports the principles of Def(Aust) 5679. Clearly such a tool should address the shortcomings of Cassandra, therefore it was decided that Cassandra's shortcomings would become functional requirements for the tool.

## 4 Example: fuel storage room

We will illustrate the tool with an example. The example concerns the hazard management for a hypothetical fuel storage room. The storage room is designed to hold drums containing petrol. To mitigate against the build up of flammable vapour the room is fitted with an exhaust fan. The room also includes a sensor for detecting the presence of flammable vapours and an alarm for warning of a dangerous build up of vapour. The exhaust fan and alarm are activated by a shared software component that takes inputs from the sensor. The software can be programmed by setting a tolerance level with the regard to the amount of vapour that is required before the alarm and exhaust fan are activated. Fig. 1 contains a data flow diagram for the storage room system.

The main accident related to the system is an explosion caused by a match being lit in the vicinity of the storage room. We differentiate between accidents leading to a single fatality and those leading to multiple fatalities.

**Acc1:** Person killed in explosion [*fatal*]

**Acc2:** Multiple deaths in explosion [*catastrophic*]

We focus on one system hazard that can lead to these accidents, given below together with the corresponding system safety requirement:

**SH1:** Petrol vapour build up in store room.

**SSR1:** Prevent petrol vapour build up.

We also identify a number of co-effectors with the probability of occurence, which appear in accident sequences from system hazards to accidents:

**Coeff1:** Someone lights a match [probability 0.05]

**Coeff2:** Petrol vapour escapes from drums [probability 0.1]

**Coeff3:** Multiple personnel in area [probability 0.1]

There are two accident sequences that lead from the system hazard SH1 to the accidents Acc1 and Acc2 respectively. The first accident sequence involves the first and second co-effectors and leads to Acc1 (single fatality). The second accident sequence involves all three coeffectors and leads to Acc2 (multiple fatalities).

Next we need to decompose the system hazard into component hazards. We use fault tree analysis (Leveson 1995, Storey 1996) to do this decomposition. The resulting fault tree is shown in Fig. 2, with the system level hazard (SH1) at the root, and the component level hazards at the leaves.

The component level hazards in the fault tree are summarised as follows:

**CH1:** Exhaust fan not working or ineffective.

**CH2:** Operator sets vapour detection level too high.

**CH3:** Vapour sensor fails.

**CH4:** Software incorrect.

**CH5:** Alarm fails.

From these component level hazards the following component level safety requirements can be derived:

**CSR1:** Store room shall have an exhaust fan.

**CSR2:** Operator shall set vapour detection level appropriate for room.

**CSR3:** Sensor shall measure vapour levels to within 5% accuracy.

**CSR4:** System shall activate fan and alarm when vapour levels exceed level set by user.

**CSR5:** Alarm shall be connected to guard room.

## 5 Conceptual design

*HazLog* was designed by doing a detailed analysis of Def(AUST) 5679, and defining use cases for the proposed tool based on information contained in the standard. This enabled us to identify the main objects to be used and manipulated by the tool. The next step involved identifying the relationships and multiplicities between these objects based on information in the standard. This information is captured using an association diagram, shown in Figure 3.

In the association diagram the (solid) lines between each object represent relationships. Multiplicities for these relationships are given, where $1 \mathinner{..} 1$ means exactly 1, $1 \mathinner{..} *$ means one or more, and $*$ means zero or more. So for example, the association diagram defines a relationship between LOT objects and SSR objects, such that:

- each LOT object is related to zero or more SSR objects, and
- each SSR object is related to exactly one LOT object (i.e., a SSR must be allocated a LOT but cannot be allocated multiple LOTs).

In this section we will describe the main objects used in the tool, and the interrelationships between these objects. We include screenshots that illustrate how information is represented in the tool (using modules).

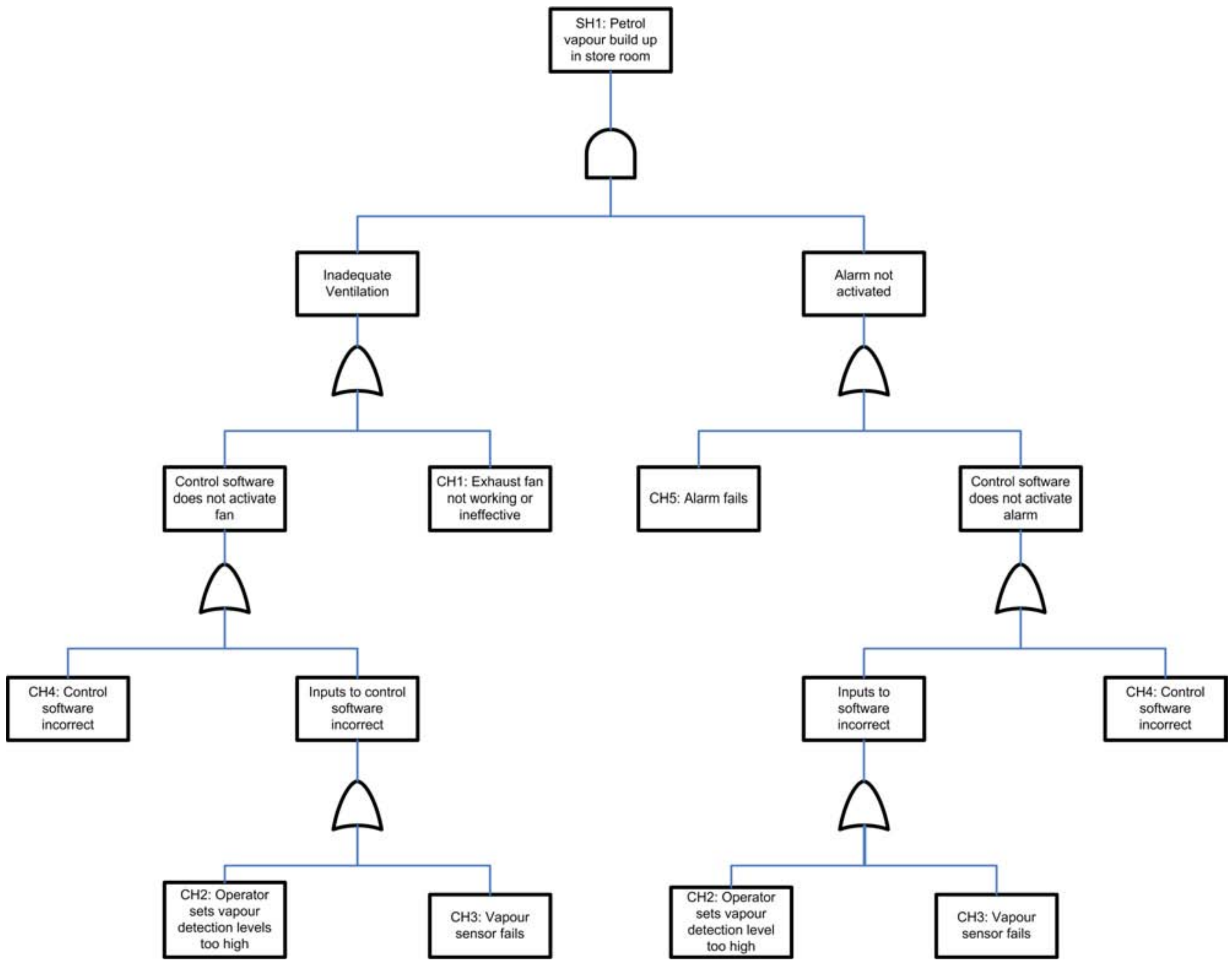

SH1: Petrol vapour build up in store room
Inadequate Ventilation
Alarm not activated
Control software does not activate fan
CH1: Exhaust fan not working or ineffective
CH5: Alarm fails
Control software does not activate alarm
CH4: Control software incorrect
Inputs to control software incorrect
Inputs to software incorrect
CH4: Control software incorrect
CH2: Operator sets vapour detection levels too high
CH3: Vapour sensor fails
CH2: Operator sets vapour detection level too high
CH3: Vapour sensor fails

Figure 2: Fault tree fuel store room

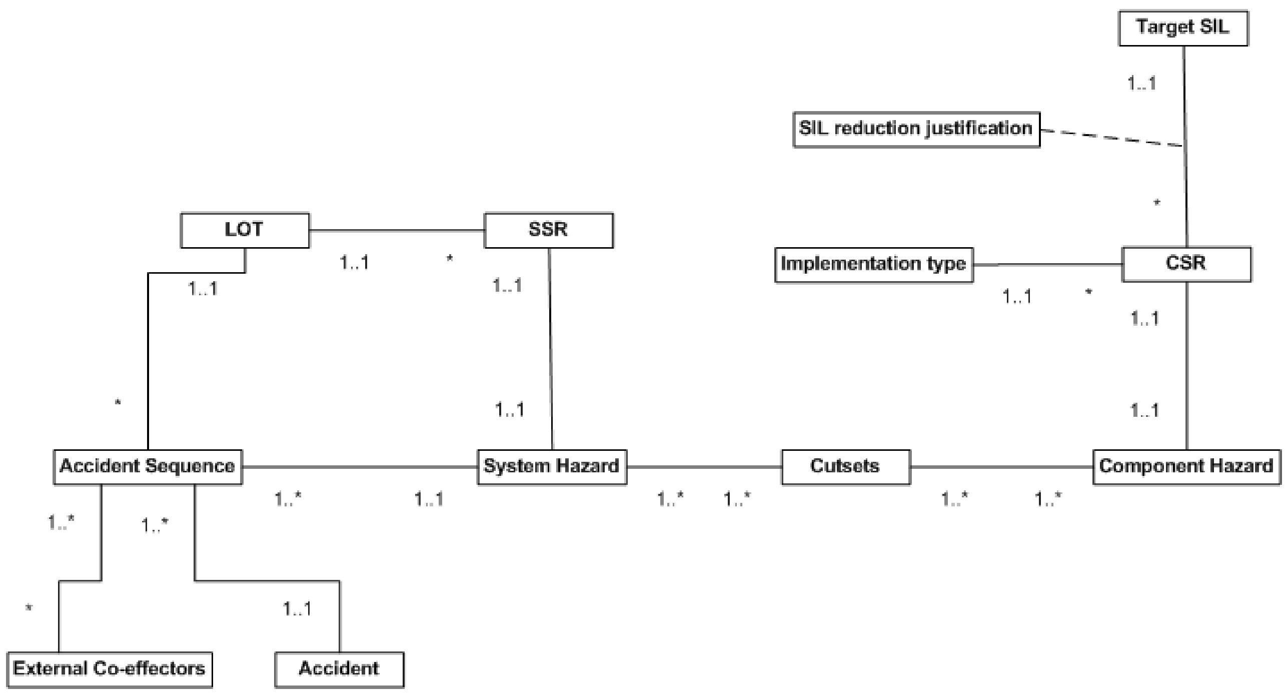
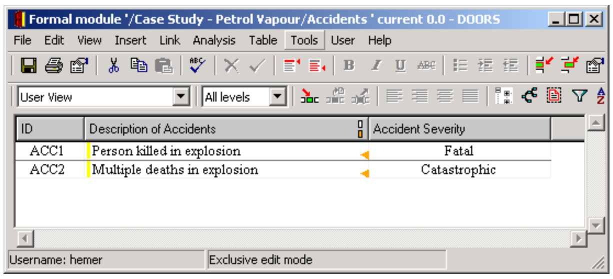


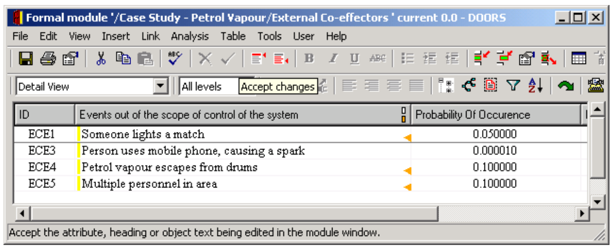
Figure 3: Conceptual model


Formal module '/Case Study - Petrol Vapour/Accidents ' current 0.0 - DOORS

| ID | Description of Accidents | Accident Severity |
|---|---|---|
| ACC1 | Person killed in explosion | Fatal |
| ACC2 | Multiple deaths in explosion | Catastrophic |

Username: hemer
Exclusive edit mode

Figure 4: Accident module


Formal module '/Case Study - Petrol Vapour/External Co-effectors ' current 0.0 - DOORS

| ID | Events out of the scope of control of the system | Probability Of Occurence |
|---|---|---|
| ECE1 | Someone lights a match | 0.050000 |
| ECE3 | Person uses mobile phone, causing a spark | 0.000010 |
| ECE4 | Petrol vapour escapes from drums | 0.100000 |
| ECE5 | Multiple personnel in area | 0.100000 |

Accept the attribute, heading or object text being edited in the module window.

Figure 5: External co-effectors

### 5.1 Accidents and co-effectors

The two simplest objects in the tool are *accidents* and *co-effectors*. Each are represented by a module with no outgoing links. Fig. 4 shows the accidents module. Each accident is given a unique identifier, an accident description and a severity. The identifier field is automatically generated by DOORS (as are the identifiers used in the other modules). The accident description field in entered as text by the user. The severity is represented as an enumerated type with the values: catastrophic; fatal; severe; or minor. This field is selected by the user in *HazLog* from a pull down menu. Fig. 5 shows the external co-effectors module in *HazLog* for the fuel storage room case study. Each external co-effector has a unique identifier, a description of the event, and the likelihood of the event occuring. The description is entered by the user. The likelihood is expressed as a probability between 0 and 1. This probability is represented internally as a decimal number with 6 decimal places, while any smaller number will be rounded to zero. If the probability is omitted it will default to 1. This user enters a value for this field, with *HazLog* automatically rounding when necessary.

### 5.2 Accident sequences

An accident sequence is modelled as a link between an accident, a system hazard and a set of external co-effectors. Each external co-effector may have an associated probability. If the probability is omitted, a default value of 1 is assumed. Before accident sequences can be created we must first add accidents and external co-effectors to the database.

Fig. 6 shows the accident sequences module for our case study. Each accident sequence consists of an identifier, a link to exactly one system hazard, a link to zero or more external co-effectors, and a link to exactly one contributing accident. The accident sequence descriptor is entered by the user. The links are created by the user in *HazLog*. The system hazard link is created by linking to an object in the system level module. Link creation in *HazLog* is based on the linking mechanism provided by DOORS. It involves creating the start of the link (in this case the hazard field) and then creating the end of the link (in this case a system level object in the system level module, shown in Fig. 7). In the case of external co-effectors, multiple links may be created. For each accident sequence an auxiliary LOT is also calculated. This auxiliary LOT will be used in calculating a LOT for each of the system safety requirements. This LOT is calculated automatically by *HazLog*.

For each system safety requirement we calculate the desired LOT. LOTs are calculated by looking at the accident sequences involving the associated system hazard. For each accident sequence we calculate an auxiliary LOT. This is a combination of the accident severity, and the probability that the system hazard leads to the accident sequence.

To calculate this probability multiply the probabilities of the external co-effectors involved in the accident sequence. We assume that the external co-effectors are independent and that all co-effectors must occur for the accident to arise. In an earlier version of the tool we used rich traceability to allow other logical combinations of external co-effectors. The meant for example we could model the case where alternative external co-effectors can lead to the accident arising. However it was decided that this level of genericity was not required in the context of Def(Aust) 5679. After calculating auxiliary LOTs, the system safety requirement LOT becomes the highest auxiliary LOT.

### 5.3 System-level hazards

The system level module captures information about system level objects, including hazards and their corresponding system safety requirements. Fig. 7 shows the system level module for our case study.

There is a 1-1 relationship between system hazards and system safety requirements. This multiplicity is enforced in the tool by ensuring that every entry in the system level module has exactly one hazard and exactly one system safety requirement. Each of these fields is entered by the user of *HazLog*.

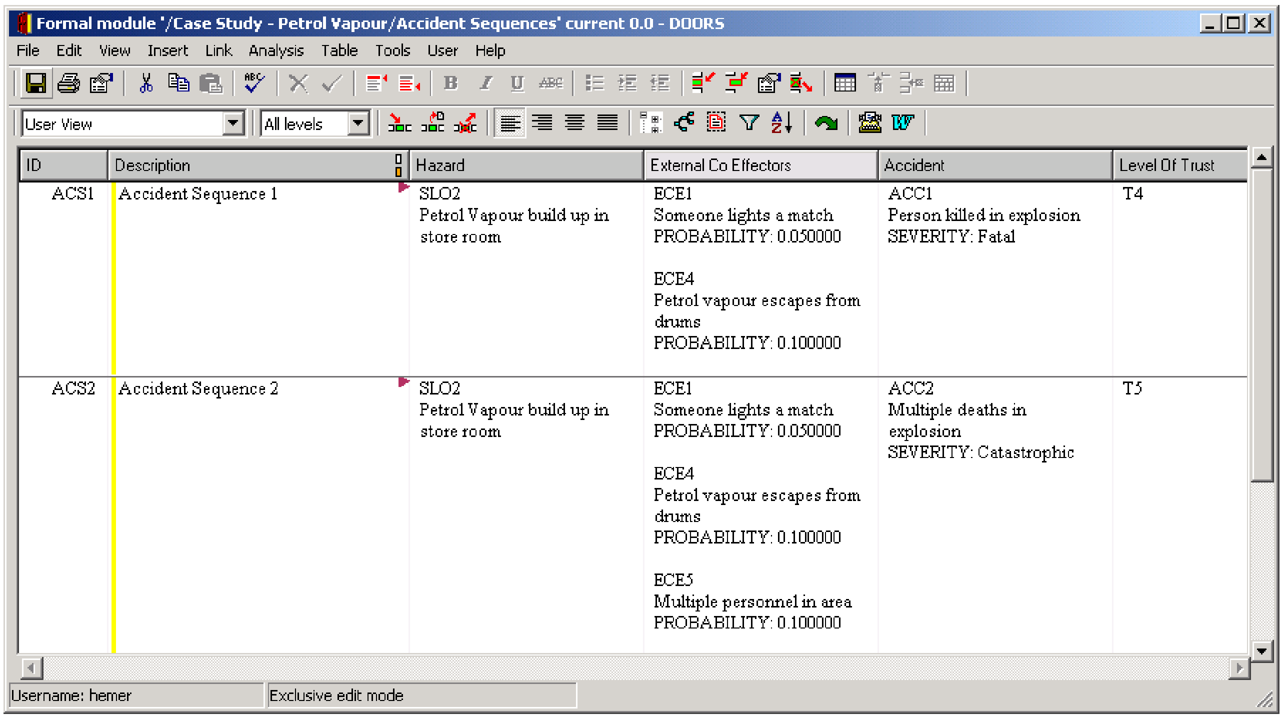
Each system hazard may be involved in one or more accident sequences (we only consider system hazards that can lead to an accident). All associated accident sequences are shown in the system level object module as links, together with an auxiliary LOT. Typically these links will be created within the accident sequences module, but will be displayed in this module. For each system safety requirement *HazLog* calculates a LOT. This LOT corresponds to the highest auxiliary LOT for the linked accident sequences.

The linking between system level hazards and component level hazards is done using minimal cutsets. Each system level hazard may be linked to one or more cutsets. Links to cutsets will typically be created by the user within the cutsets module (see Fig. 9). More details on minimal cutsets is given in Section 5.5.

### 5.4 Component-level hazards

Fig. 8 shows the component level objects module, used to store component hazards and associated information, for our case study. A component level hazard has a 1-1 relationship with a component safety requirement (CSR). The hazard description and CSR fields are entered by the user of *HazLog*. Each CSR is assigned a SIL by the user of the tool. The SIL is represented as an enumerated type, with the user selecting the appropriate value from a menu. In order to check the SILs that have been assigned to CSRs, each CSR has an associated implementation type. The implementation type is represented as an enumerated type with the following values:

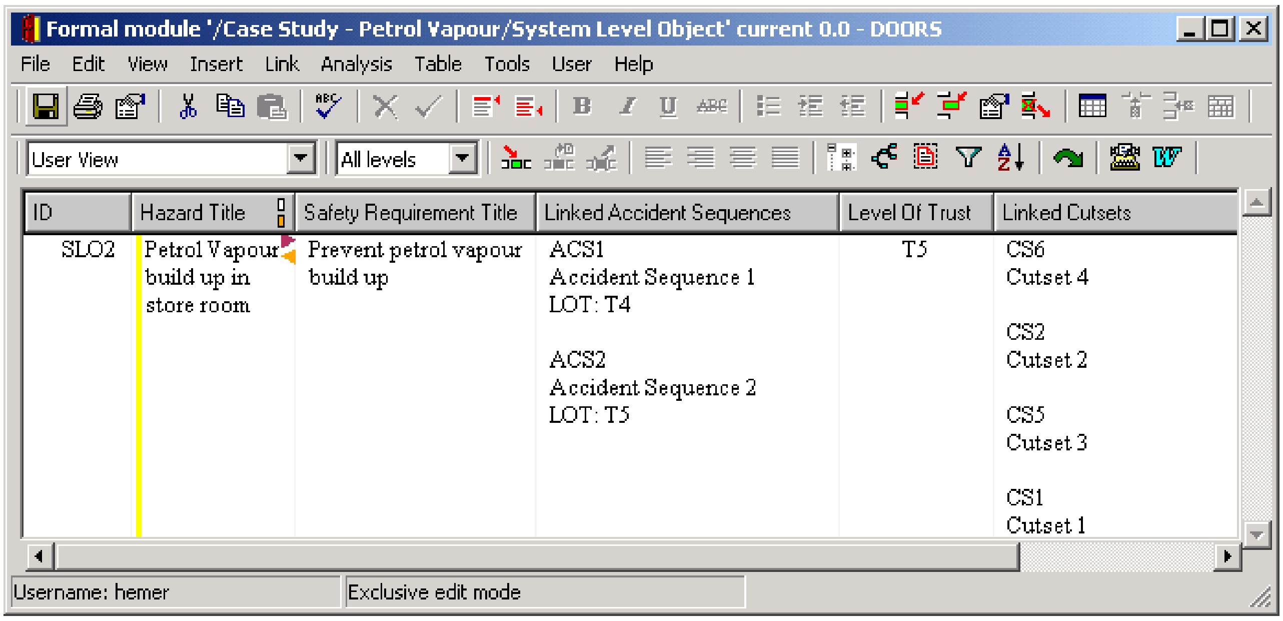
- hardware


Formal module '/Case Study - Petrol Vapour/Accident Sequences' current 0.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Help
User View
All levels

| ID | Description | Hazard | External Co Effectors | Accident | Level Of Trust |
|---|---|---|---|---|---|
| ACS1 | Accident Sequence 1 | SLO2<br>Petrol Vapour build up in store room | ECE1<br>Someone lights a match<br>PROBABILITY: 0.050000<br><br>ECE4<br>Petrol vapour escapes from drums<br>PROBABILITY: 0.100000 | ACC1<br>Person killed in explosion<br>SEVERITY: Fatal | T4 |
| ACS2 | Accident Sequence 2 | SLO2<br>Petrol Vapour build up in store room | ECE1<br>Someone lights a match<br>PROBABILITY: 0.050000<br><br>ECE4<br>Petrol vapour escapes from drums<br>PROBABILITY: 0.100000<br><br>ECE5<br>Multiple personnel in area<br>PROBABILITY: 0.100000 | ACC2<br>Multiple deaths in explosion<br>SEVERITY: Catastrophic | T5 |

Username: hemer | Exclusive edit mode

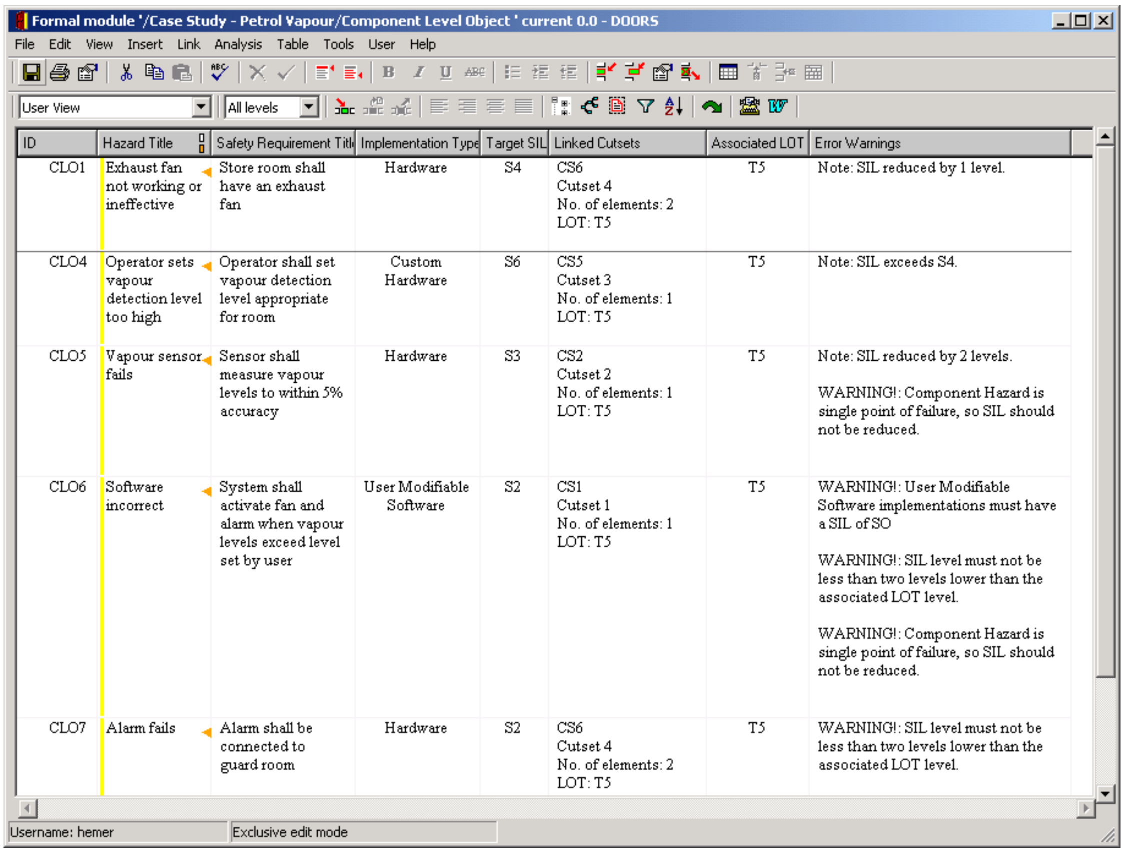
Figure 6: Accident sequences


Formal module '/Case Study - Petrol Vapour/System Level Object' current 0.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Help
User View
All levels

| ID | Hazard Title | Safety Requirement Title | Linked Accident Sequences | Level Of Trust | Linked Cutsets |
|---|---|---|---|---|---|
| SLO2 | Petrol Vapour build up in store room | Prevent petrol vapour build up | ACS1<br>Accident Sequence 1<br>LOT: T4<br><br>ACS2<br>Accident Sequence 2<br>LOT: T5 | T5 | CS6<br>Cutset 4<br><br>CS2<br>Cutset 2<br><br>CS5<br>Cutset 3<br><br>CS1<br>Cutset 1 |

Username: hemer | Exclusive edit mode

Figure 7: System-level hazards and safety requirements


Formal module '/Case Study - Petrol Vapour/Component Level Object ' current 0.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Help
User View
All levels

| ID | Hazard Title | Safety Requirement Title | Implementation Type | Target SIL | Linked Cutsets | Associated LOT | Error Warnings |
|---|---|---|---|---|---|---|---|
| CLO1 | Exhaust fan not working or ineffective | Store room shall have an exhaust fan | Hardware | S4 | CS6<br>Cutset 4<br>No. of elements: 2<br>LOT: T5 | T5 | Note: SIL reduced by 1 level. |
| CLO4 | Operator sets vapour detection level too high | Operator shall set vapour detection level appropriate for room | Custom Hardware | S6 | CS5<br>Cutset 3<br>No. of elements: 1<br>LOT: T5 | T5 | Note: SIL exceeds S4. |
| CLO5 | Vapour sensor fails | Sensor shall measure vapour levels to within 5% accuracy | Hardware | S3 | CS2<br>Cutset 2<br>No. of elements: 1<br>LOT: T5 | T5 | Note: SIL reduced by 2 levels.<br><br>WARNING!: Component Hazard is single point of failure, so SIL should not be reduced. |
| CLO6 | Software incorrect | System shall activate fan and alarm when vapour levels exceed level set by user | User Modifiable Software | S2 | CS1<br>Cutset 1<br>No. of elements: 1<br>LOT: T5 | T5 | WARNING!: User Modifiable Software implementations must have a SIL of SO<br><br>WARNING!: SIL level must not be less than two levels lower than the associated LOT level.<br><br>WARNING!: Component Hazard is single point of failure, so SIL should not be reduced. |
| CLO7 | Alarm fails | Alarm shall be connected to guard room | Hardware | S2 | CS6<br>Cutset 4<br>No. of elements: 2<br>LOT: T5 | T5 | WARNING!: SIL level must not be less than two levels lower than the associated LOT level. |

Username: hemer
Exclusive edit mode

Figure 8: Component-level hazards and safety requirements

- operator procedure
- software
- user modifiable software

The SIL and implementation types are selected by the user from a drop down menu.

In Section 6 we describe the SIL allocation checking rules that are implemented in the tool. Some of these rules use the implementation type to determine whether or not the allocation can be justified.

For each component level hazard, a list of minimal cutsets linked to this hazard is shown. Links to these cutsets are usually created within the cutsets module. This module also calculates an associated LOT for each component level hazard. This associated LOT corresponds to the highest LOT of corresponding system level hazards (related via a cutset). The associated LOT is used in checking SIL allocations (see Section 6 for more details).

The module also records any violations of SIL allocation rules. A distinction is made between warnings, where a rule has been violated directly, and a note, where further justification is usually required.

### 5.5 Linking system-level and component-level hazards

System hazard analysis involves decomposing system hazards into the set of component hazards that can result in the system hazard. Fault tree analysis is a commonly used technique for decomposing system level hazards down to component level hazards. A fault tree is a tree structured notation, with the causes of higher level events combined using logical gates.

Of interest to us are fault trees with system hazards at the root, and component hazards at the leaves. However fault trees will often have a number of intermediate levels; while this information is important we do not need to record this level of detail in the tool. Instead we use *minimal cutsets* to record the (minimal) combinations of component hazards that can by themselves lead to the system hazard.

A cutset is modelled as a set of component level hazards $\{CH_1, .., CH_n\}$. Logically, a minimal cutset leads to system hazard if and only if all of the component level hazards occur. A system level hazard may have multiple minimal cutsets, any one of these cutsets by itself can lead to the system hazard.

Consider the fault tree for our example, shown in Fig. 2. This fault tree has the following cutsets:

**Cutset1:** CH2

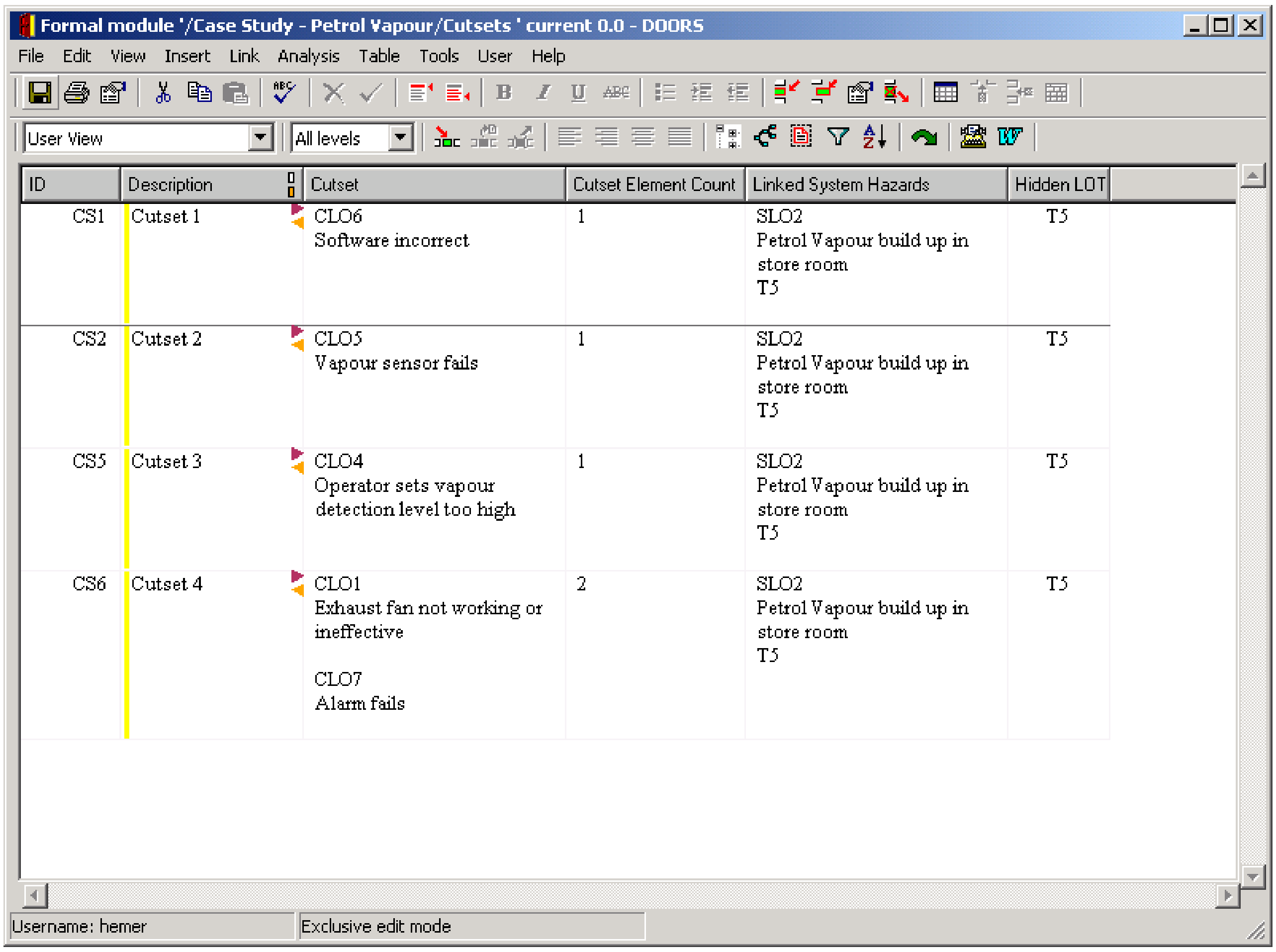
**Cutset2:** CH3

**Cutset3:** CH4

**Cutset4:** CH1 + CH5

**Aside:** Notice that in this case there are three cutsets with only one component hazard, corresponding to single points of failure (i.e., a single failure at the component level can result in a hazard at the system level). This is indicative of a poor design (sometimes single points of failure cannot be avoided, but often they can be removed with a better design). The problem in our example stems mainly from the fact that both layers of protection — the exhaust fan and the alarm — will only be activated if the control software is functioning correctly. A better design would be for the exhaust fan to be operating continually, thus being independent of the control software.

Cutsets are created in *HazLog* via the cutsets module, shown in Fig. 9. A cutset consists of an identifier (automatically created) and a descriptor. A cutset also includes a link to one or more system objects and links to one or more component-level objects. These links are created by the user, typically within the cutsets module.


Formal module '/Case Study - Petrol Vapour/Cutsets' current 0.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Help
User View
All levels

| ID | Description | Cutset | Cutset Element Count | Linked System Hazards | Hidden LOT |
|---|---|---|---|---|---|
| CS1 | Cutset 1 | CLO6<br>Software incorrect | 1 | SLO2<br>Petrol Vapour build up in store room<br>T5 | T5 |
| CS2 | Cutset 2 | CLO5<br>Vapour sensor fails | 1 | SLO2<br>Petrol Vapour build up in store room<br>T5 | T5 |
| CS5 | Cutset 3 | CLO4<br>Operator sets vapour detection level too high | 1 | SLO2<br>Petrol Vapour build up in store room<br>T5 | T5 |
| CS6 | Cutset 4 | CLO1<br>Exhaust fan not working or ineffective<br><br>CLO7<br>Alarm fails | 2 | SLO2<br>Petrol Vapour build up in store room<br>T5 | T5 |

Figure 9: Minimal cutsets

## 6 SIL allocation checks

Unlike LOTs which are calculated automatically for SSRs, based on accident sequences, SILs are assigned to CSRs by the user. However each SIL assignment must be justified, and Def(AUST) 5679 includes a number of SIL allocation rules that must be checked. Some of these checks have been implemented in the tool, as listed below. If violations of the rules are detected then a warning will be displayed.

### 6.1 Rule 1

*The SIL of a CSR shall be no less than two levels lower than the LOT of the SSR from which it was derived* (DEF(Aust) 5679 1998, paragraph 15.4.9).

We assume that a CSR can be linked to multiple SSR, however it is sufficient in this case to just use the highest LOT.

To check the rule, *HazLog* maintains a set of linked cutsets for each component object (for example, see column 7 of the component level object module shown in Fig. 8). This set corresponds to all cutsets that include the component level hazard. For each linked cutset, *HazLog* also records the number of elements in the cutset, and the LOT of the corresponding system level hazard. An "associated LOT" (see column 8 of the component level object module shown in Fig. 8), is then calculated for each component object by taking the *highest* LOT for the linked cutsets. *HazLog* then calculates the difference between the target SIL and the associated LOT, issuing a warning whenever the rule is violated, and noting any other reductions (i.e., where the SIL is lower than the associated LOT).

An example of a violation of this rule in the case study, is the component-level hazard CH4 "Software incorrect", which has been assigned a target SIL of $S2$. In this case, *HazLog* determines that the associated LOT is $T5$, and therefore the target SIL has been reduced by three levels.

### 6.2 Rule 2

*Software that can be modified by a user after installation shall be assigned a SIL of S0* (DEF(Aust) 5679 1998, paragraph 15.4.12).

The check for this rule is straightforward. *HazLog* checks any CSR with a *user modifiable software* implementation type. If the SIL of the CSR is greater than S0, then a warning is raised.

An example of such a violation in the case study is for the fourth safety requirement shown in Fig. 8, i.e.,

> System shall activate fan and alarm when vapour levels exceed level set by the user.

We set the implementation type to user modifiable software for this CSR because the software can be modified by the user to adjust the tolerance levels. However the target SIL has been set to S2 resulting in a clear violation of the rule.

### 6.3 Rule 3

*The SIL assigned to a Component to be implemented by an operator procedure should not normally be higher than S3* (DEF(Aust) 5679 1998, paragraph 15.4.11).

The check for this rule is similar to that of Rule 2. In this case *HazLog* checks the SIL for any CSR with an *operator procedure* implementation type. For any such CSRs with a SIL greater than S3, a warning is raised.

An example violation in the case study is for the second CSR shown in Fig. 8, i.e.,

> Operator shall set vapour detection level appropriate for room.

For this CSR we have set the implementation type to operator procedure, but the allocated SIL is S4, again a clear violation of the rule.

### 6.4 Rule 4

*The SIL of a CSR for a component belonging to a singleton cutset shall not be less than the LOT of the SSR from which it was derived*

This rule is checked by looking at all component hazards that belong to a minimal cutset with only one element. For each such component hazard, the corresponding SIL is compared against the highest LOT associated with the system hazards that are related to the cutset (recall that a cutset may be related to more than one system hazard). For any SIL that has been reduced below the LOT, a warning is raised.

The case study includes three minimal cutsets with only one component, linked to the second, third and fourth component hazards (single points of failure). For each of these hazards the associated SIL has been reduced below the LOT for the corresponding SSR (T5 in each case).

## 7 Discussion

*HazLog* is a prototype implementation of the conceptual design described in the previous sections. The implementation is based on DOORS, and consists of a collection of modules (initially empty), links between these modules, and code for performing calculations (e.g., LOT calculation) and data integrity checks (e.g., SIL allocation checks). DOORS has proven to be a good basis for the tool, since it allows us to easily track changes to the data and link objects. It also allows for multiple views which is useful for presenting information. The current implementation supports the recording and checking of most of the safety case data for Def(Aust) 5679 as described in Part 3 of the standard.

In terms of the shortcomings of Cassandra with respect to its support for Def(Aust) 5679 (listed in Section 3) all of the points have been addressed. We briefly discuss each point:

1. *Lack of support for representing accident sequences.*

   *HazLog* allows accident sequences to be modelled as a link between an accident, a hazard and multiple co-effectors. Accidents and hazards may appear in number of accident sequences.

2. *The inability to record relationships between hazards.*

   *HazLog* allows system hazards and component-level hazards to be linked via minimal cutsets. This means that results produced from fault tree analysis can easily be represented in *HazLog.*

3. *The risk classification matrix (how risk control levels are calculated) provided by Cassandra would need considerable modifications to accommodate the Def(Aust) 5679 LOT/SIL dichotomy.*

   *HazLog* provides full support for the LOT calculation for system safety requirements, and SIL allocation for component-level safety requirements. Many of the requirements for SIL allocation listed in Def(Aust) 5679 are checked automatically by *HazLog*, with warnings raised for violations.

4. *The limited support for recording safety requirements.*

   *HazLog* allows the user to record safety requirements for each system hazard and component-level hazard.

5. *The limited search facilities through the hazard, accident and control database.*

   *HazLog* makes use of the search facilities provided by DOORS, which allows textual searches to be performed within modules.

In designing *HazLog*, we made some simplifying assumptions about safety requirements and hazards. In particular we assume that for every CSR there is at least one corresponding SSR, and that every CSR corresponds uniquely to a component-level hazard. This differs slightly from the standard which allows CSRs to be derived that do not have any corresponding component-level hazard, and do not have a corresponding SSR. (An example is the use, or non-use, of a particular programming language.) However, since it was not clear what analysis could be done on such CSRs, it was decided that this was not a major problem. Instead it was felt that Def(Aust) 5679 should better clarify exactly how such CSRs are supposed to be analysed and addressed.

*HazLog* is still in the early prototype stage, and is obviously not as mature as Cassandra. There are a number of enhancements that could be made to the tool, many based on Cassandra's capabilities, including:

- Report generation, capable of summarising for example SIL allocation violations. Report generation should be customisable by the user, with the ability to focus on certain aspects or certain pieces of data.
- Status control on hazard data, indicating for example whether data is in draft form, or has been reviewed, or whether a particular hazard has been adequately mitigated.
- Access control on the data base, so that (read/write) access to data can be restricted to authorised personnel.
- Links to supporting documents. At this stage *HazLog* supports text fields for SIL justification for example. This should be updated to allow links to supporting evidence, containing detailed rationale for the SIL allocation.
- A forms based user interface, allowing a more intuitive entry of data. A graphical user interface would be useful for accident sequences, for example.

## 8 Conclusions

The development of safety cases involves recording and analysing large amounts of data with complex interrelationships. In managing this data it is important to be able to determine whether all hazardous aspects of the system have been adequately addressed. Without tool support this can be a difficult task, in which weak points in the safety case can be easily overlooked. Existing tools, in particular Cassandra, are an excellent aide in hazard management, with the ability to tailor the tool to suit processes from a variety of safety standards.

However Def(Aust) 5679, a standard used by the Australian Defence forces for the development of safety critical computer-based systems, differs in a number of fundamental areas in comparison to other standards. For this reason, Cassandra has proven to be unsuitable for supporting the Def(Aust) 5679 process.

In light of the shortcomings of Cassandra, we have designed and implemented a prototype tool, *HazLog*, for supporting hazard management within the Def(Aust) 5679 process. *HazLog* supports the recording and analysis of accident, hazard and safety requirement data. Through a simple example, we have illustrated the capabilities of *HazLog*, and shown that it overcomes the shortcomings of Cassandra with respect to support for Def(Aust) 5679. However we do note that *HazLog* is still in the prototype stage, and we have listed a number of possible enhancements that could be made to the tool.

Contact the last author for more details of the tool and its status.

## References

DEF(Aust) 5679 (1998), *Procurement of Computer-Based Safety Critical Systems*, Land Engineering Agency, Australian Department of Defence. http://www.dsto.defence.gov.au/isl/defaust5679.html.

HVR (2000), *Cassandra User Guide Version 3.0*, HVR Consulting Services Ltd. http://www.hvr-csl.co.uk/products/cassandra.html.

IEC 61508 (1998), 'Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems', International Standard IEC 61508.

Leveson, N. (1995), *Safeware: System Safety and Computers*, Addison Wesley.

MIL-STD 882C (1993), *System safety program requirements*, U.S. Dept of Defense.

MOD 00-55 (1997), *Requirements for Safety-Related Electronic Hardware in Defence Equipment*, U.K. Ministry of Defence. http://www.dstan.mod.uk/ or http://www.seasys.demon.co.uk/.

MOD 00-56 (1996), *Safety Management Requirements for Defence Systems*, U.K. Ministry of Defence. http://www.dstan.mod.uk/ or http://www.seasys.demon.co.uk/.

Storey, N. (1996), *Safety-Critical Computer Systems*, Prentice Hall.

SVRC Services (2001), Evaluation of Cassandra Hazard Management Tool Version 3.01a Issue 1.1, Department of Defence Document ID:CA38809-212.

Telelogic DOORS (2004). http://www.telelogic.com/doors/.

Wabenhorst, A. & Atchison, B. (1999), A survey of international safety standards, Technical Report 99-30, Software Verification Research Centre, School of Information Technology, The University of Queensland, Brisbane 4072, Australia. http://www.itee.uq.edu.au/~defsafe/Publications/svrc1999-30.