Probabilistic Model-Checking Support for FMEA

Lars Grunske Robert Colvin Kirsten Winter
ARC Centre for Complex Systems
School of Information Technology and Electrical Engineering
University of Queensland
4072 Brisbane, Australia
{grunske,robert,kirsten}@itee.uq.edu.au

Abstract

Failure Mode and Effect Analysis (FMEA) is a method for assessing cause-consequence relations between component faults and hazards that may occur during the lifetime of a system. The analysis is typically time intensive and informal, and for this reason FMEA has been extended with traditional model checking support. Such support does not take into account the probabilities associated with a component fault occurring, yet such information is crucial to developing hazard reduction strategies for a system. In this paper we propose a method for FMEA which makes use of probabilistic fault injection and probabilistic model checking. Based on this approach safety engineers are able to formally identify if a failure mode occurs with a probability higher than its tolerable hazard rate.

Keywords: Probabilistic Model Checking, Failure Mode and Effect Analysis, System Safety

1. Introduction

One of the most important goals when developing a safety critical system is to eliminate potential hazards that can lead to accidents that cause injuries or the loss of life. In reality, because components can fail over time, it is impossible to create a physical system that is completely hazard-free. Safety analysis techniques therefore focus on minimising risk, by, for example, installing more reliable components, having back-up systems, or regular maintenance and replacement of system components.

A well-known technique for safety evaluation is Failure Mode and Effect Analysis (FMEA) [29], where a team of engineers identify the effects and cause-consequence relationships of component failures on system-level hazards. Once a cause-consequence relationship is detected between a failure mode and a hazard, the likelihood of that hazard occurring is calculated and compared to its tolerable hazard rate. If the system does not pose a non-tolerable risk then the system is considered safe, otherwise steps must be taken to reduce the probability of the hazard occurring. Traditionally, the FMEA process is labour-intensive and performed by-hand. The hazards that a component fault can cause are determined informally, and the probability of the failure modes occurring under those circumstances are calculated by consulting data from component specifications and previous experience. To assist the FMEA process, model checking has been used to automatically determine whether a hazard will occur in presence of a given failure mode of a system component [7, 10, 22, 27]. While this approach can help to identify critical relationships between component failure modes and hazards in a system, because standard model checking is non-probabilistic, it does not address the probability with which a failure mode occurs and consequently it is not possible to determine the probability with which a hazard violation may occur. Therefore, a major part of the FMEA process is still left as a non-automated and error-prone task.

The contribution of this paper is to provide a methodology for FMEA which incorporates probabilistic modelling and model checking. We will refer to this methodology as probabilistic FMEA (pFMEA). It has the advantage of formally including rates at which component failures can occur, allowing the rate at which a hazard can occur to be formally calculated via probabilistic analysis. If a hazard is found to occur at too high a rate, model checking tools may be used to experiment with different versions of the system model, such as with more reliable components or faster repair rates. To use probabilistic model checking with FMEA, the system model, including its interaction with the environment, must be written probabilistically, e.g., as a continuous time Markov chain, Markov decision process, etc. The rate at which a component fails is built into the models of faulty behaviour, and the tolerable hazard rate is associated with the probabilistic properties which represent the safety requirements, which are checked against the model.
The paper is organised as follows: Section 2 introduces the preliminaries of system hazard and risk analysis, as well as details about FMEA. In Section 3 we introduce a methodology to support FMEA with probabilistic model checking. We demonstrate the method with a running example. Related work is summarised in Section 4 and we conclude with an outlook to future work in Section 5.

2 Preliminaries

In this section we summarise the basic terminology and principles of safety analysis processes, and describe Failure Mode and Effect Analysis (FMEA), a hazard analysis technique, which is central to this paper.

2.1 Safety Analysis

A safety analysis process is commonly structured into two separate sub processes, that are called risk analysis and hazard analysis [21, 39, 40, 44], as shown in Figure 1. The aim of risk analysis is to identify potential hazards that could occur during the system’s lifetime and to determine their tolerable hazard probabilities or rates (THR) (the mid-point in Figure 1). A hazard is defined [21, 39] as a state of a system and its environment in which the occurrence of an accident which may result in an injury or death only depends on factors that are not under control of the system. For example a hazard of a level crossing system is when the gates are open and a train is in the critical section of the level crossing – an accident occurring depends only on drivers and cars that may cross the level crossing section. Consequently, it is the goal of safety engineers to reduce the probability of a hazard occurring, since effects outside the system boundary are not under the control of the engineer. To determine the THR, it is necessary to quantify the risk that is caused by the hazards of the system, which is the combination of the severity of possible accidents and the probability that a hazard will lead to this accident. Whether a risk is acceptable may depend on social factors, such as applicable laws or public opinion. For practical reasons, standards (e.g. [9, 32]) have defined risk acceptance criteria such as ALARP (As Low As Reasonably Practicable), GAMAB (Globalement Au Moins Aussi Bon) and MEM (Minimum Endogenous Mortality). Based on the risk acceptance criteria, the severity of possible accidents and the probabilities that the hazard will lead to this risk, a THR can be assigned for each of the hazards.

The link between the risk analysis and the following hazard analysis process is a document that contains the identified safety requirement. In our context, a safety requirement states that a specific hazard should not occur with a probability higher then its THR [39, 21]. The goal of the risk analysis process is the formal definition of these safety requirements.

Hazard analysis is used to evaluate whether a system design meets its safety requirements. The main evaluation can be distinguished into two phases. In the first phase, explorative methods are used to identify component failure modes. A component failure mode represents the behaviour of the component when it has (partially or completely) failed – for example, when a sensor outputs incorrect data. Once the failure modes and hazards have been identified, cause-consequence relationships between them must be determined. The initial identification of failure modes is typically based on expert knowledge, although checklists or fault specifications for common components may also be used. To identify further relevant hazards and component failure modes, the exploration is conducted as alternating forward and backward searches [41]. The backward search attempts to identify all relevant causes for each hazard, and is commonly performed using Fault Tree Analysis (FTA) [47, 30]. The forward search identifies the consequences of already identified fault modes, and is commonly performed by a systematic technique known as Failure Modes and Effects Analysis (FMEA) [29]. FMEA is the focus of this paper, and is discussed in more detail below. In practice, both forward and backward searches are performed by a team

![Figure 1. Risk and Hazard Analysis Process](image-url)
of expert analysts, and the result of the process is an evaluation model that clearly specifies the cause-consequence relationship between low-level failure modes and system level hazards. Commonly, basic fault trees are used as a formal representation of the cause consequence relationships. However, currently there is also a trend towards state based models, such as Markov models [31], Stochastic Petri Nets [42] State Event Fault Trees [20] or Dynamic Fault Trees [17], since these models can describe the temporal order in which failures must occur in order to cause a hazard.

In the second phase of hazard analysis, the probabilities of hazards are determined based on the evaluation model using analytical, numerical or simulative methods. If the probability that the hazard occurs during the lifetime of the system is lower than its THR then the system fulfills the safety requirement. The results of a successful safety evaluation can be used to document the steps the company has taken to comply with safety regulations. If the safety evaluation is unsuccessful, the design must be revised by introducing safety improving mechanisms, such as watchdogs, redundancy etc., to identify and mask failure modes and mitigate the probability that a hazard can occur.

2.2 Failure Mode and Effect Analysis

The aim of FMEA is to systematically identify failure modes of system components and to explore if these failure modes will lead to potential hazards.Traditionally, FMEA is a team-based exercise, where the team tries to predict how a failure mode of a component will effect the correct operation of the system. Of special importance are deviations from the correct behaviour that may lead to system hazards. If a relationship between a component failure mode and a system hazard is identified, the failure mode has safety critical implications and the FMEA team must identify details of this relationship and all co-effectors that are required. Additionally, the team tries to identify measures for reducing the probability of occurrence of the failure mode and the propagation of resulting failures. The overall process is repeated for each failure mode of all relevant components in the system. During the FMEA process, the results of the investigation are documented in a table, which specifies for each component the set of relevant component failure modes and their consequences in terms of possible hazards. Additionally, the table documents possible failure detection, and correction or mitigation mechanisms may be recommended. The structure, number of columns and meaning of columns of the resulting FMEA table may vary in different companies, though typically the following column headings are used [29]: investigated component; failure mode; description of the failure mode/local effect of the failure mode; possible cause for the failure; effect at the system level; recommended failure detection mechanism; recommended mitigation mechanism; and recommended design changes.

FMEA can be supported by model checking and fault-injection experiments [8, 10, 11, 22, 27, 45]: If the system model (in normal as well as failure mode) is provided in a formal notation that can be input to a model checker and the safety conditions are specified as temporal logic formulas, then the model checker can check if in a particular failure mode the safety conditions are violated. If a violation occurs (depending on the tool in use) a counter-example will be provided showing one possible behaviour that leads to the violation of the safety condition, namely the hazardous state. For the process of FMEA, this counter-example can provide useful insight into the relationship between component failures, hazard and co-effectors.

3. Probabilistic FMEA

Probabilistic FMEA (pFMEA) builds on previous work using standard model checking to support the FMEA process. The user specifies probabilistic system behaviour such as failure occurrence as well as the tolerable rates for hazards to occur. We may therefore ask for the likelihood with which hazards can occur when the system is in a particular failure mode. This benefit renders the approach interesting for the concerns of FMEA when compared to existing techniques.

![Figure 2. Probabilistic FMEA](image)

Figure 2 depicts the process of FMEA using a probabilistic model checker. The user provides a probabilistic model of the system in normal behaviour, $F_{V0}$, as well as a set of so called failure views, $F_{V1}, ... , F_{Vn}$. A failure view is a model of the system exercising a particular component failure mode. It is generated by injecting a specific fault into a system component, which results in a failure mode of this component. Additionally, the user provides the failure view matrix which specifies possible transitions including their
transition rates between the normal operation of the system and the given failure views. A failure rate, $\lambda$, specifies the rate at which a failure might occur, whereas a recovery rate, $\mu$, specifies the rate at which the system “recovers”, i.e., the failure is fixed.

The model $F_{V_0}$, together with (one or more) failure views $F_{V_i}$, and the corresponding failure view matrix, build the probabilistic model $M$ for one fault-injection experiment. From the safety requirements of the system and their tolerable hazard rates (THR) the user formalises the properties to be checked against $M$ in the experiment. That is, in an experiment we analyse the impact of one or more chosen component failure modes, that may occur with a failure rate $\lambda$ and may be fixed with a recovery rate $\mu$, to the behaviour of the system model with respect to the given safety requirements. We may then formally determine whether a violation of the safety requirements happens with a higher or lower rate than the given THR.

In the remainder of this section we describe how the inputs to the pFMEA process can be developed, and the consequences this has on the process and its results. Our presentation is given in broad terms, with the intention that any sufficiently powerful probabilistic modelling language can be used, though for concreteness we use continuous time Markov chains (CTMCs) (as surveyed in [34]). Other suitable formalisms include Markov decision processes [15] and Probabilistic Timed Automata [37], which are appropriate if the system model contains non-deterministic as well as probabilistic behaviour, and time constraints that can be expressed via clock variables, respectively.

*Example* As a running example throughout the paper we use an industrial metal press [1, 43]. The system compresses metal sheets into body parts for vehicles. The system has a motor which causes a heavy plunger to rise. When the motor is on, the operator may load a metal sheet into the press. When the plunger is at the top of the press, the operator can then push a button, which turns the motor off, causing the plunger to fall. When the plunger reaches the bottom, thereby compressing the metal, the motor is automatically turned on by the controller. The plunger begins rising again, repeating the cycle. The operator may abort the fall of the plunger by releasing the button, but only if the plunger has not yet reached the “point of no return” (PONR). When the plunger has fallen beyond the PONR it is dangerous to turn the motor on as the momentum of the plunger against the action of the motor can cause damage to the equipment and the environment. Moreover, as the plunger is rising the motor must not be switched off since the operator might still be loading the press with a new sheet and therefore be injured if the press falls.

### 3.1 Probabilistic FMEA inputs

As shown in Figure 2, the inputs to pFMEA are a model of the system, a set of hazards, a failure view matrix, and a set of component failure modes that may occur. We describe the form each of these must take for probabilistic analysis.

**System model.** The system model for pFMEA will differ from the system model used in typical formal FMEA in that the system shows probabilistic behaviour. That is, the model will specify the rate at which interactions with the environment occur. Within the system itself there may be no probabilistic behaviour, unless explicitly required for the particular system.

*Example* The model of the metal press consists of eight communicating modules: four sensor modules (TopSensor, PONRSensor, BottomSensor, ButtonSensor), the Motor, the Controller, the Plunger and the Operator. The Plunger and Operator represent the environment and are therefore only partly controlled by the system.

![Figure 3. The metal press system](image)

Figure 3 depicts the system structure and the communication between the modules: the Plunger reports its current position to the sensors which in turn communicate the change in position to the Controller. Depending on the current sensor values and the Operator’s behaviour (i.e., the button being pushed or released) the Controller sends a signal to the Motor to be turned on or off. The motor value triggers the behaviour of the Plunger, causing it to rise or fall.

*Figure 4 depicts the Motor module. Initially, the module is in state off. When a message is received (from the Controller) to turn on the motor (turnOn?) the module transitions into state switchOn from which a message is passed to the Plunger indicating that the motor is on (mOn!). When in

---

1. We decorate synchronisation events with ? and ! to indicate inputs or outputs of a module, respectively.
state on the module awaits a signal turnOff? (from the Controller) in order to transition to state switchOff. In the next step the Plunger is informed that the motor is off (mOff?).

![Figure 4. The Motor module](image)

Figure 4. The Motor module

In failure view mode the TopSensor component where it becomes stuck high and therefore never reports that the Plunger has left the top position. The normal behaviour of the TopSensor is shown in Figure 6 on the left. Initially the module is in state 0 awaiting the input atTop? from the Plunger. When this is received a message is output to the Controller indicating the plunger’s new position. When the Plunger has left the top position, the message notAtTop? is received and the Controller is notified by sending tsLow!.

In failure view mode the TopSensor will not send the message tsLow! to the Controller. The transition from state 3 to 0 becomes guarded: if in normal mode \(fv = 0\) then the message is sent as usual, else if in failure mode \(fv = 1\) the message is not sent. We use the notation guard/action in Figure 6 to model this guarded behaviour, i.e., \(fv = 0/\text{tsLow}!\) and \(fv = 1/\text{skip}\), respectively.

Note that the TopSensor with injected failure covers the behaviour for both normal and failure mode. This allows switching between the failure modes during runtime, and especially for a recovery of the fault.

Figure 5 visualises the behaviour of the Plunger module. Initially, the plunger is at the bottom and may receive the signal that the motor is switched on (mOn?) which causes it to rise, first below the PONR and then above the PONR. When stepping from state to state the module sends out messages to the respective sensors (notAtBottom?, abovePONR?, atTop!, etc.). When at the top the plunger awaits a signal that the motor is switched off before it starts falling, first above and then below the PONR. In both states of the rising phase the plunger may receive a signal that the motor is switched off (mOff?). Similarly, in both falling states the signal for the motor being switched on (mOn?) might be received. From these four transitions that change the plunger from rising to falling and vice versa without reaching top and bottom, respectively, only the transition from state fallingAbove to state risingAbove models a safe operation of the system.

In order to model a realistic timing of the interaction between the modules, each transition of the plunger is given an associated rate. For example, when raising the rate for updating the position is set to \(1/15\text{sec}^{-1}\). When falling above the PONR the plunger falls relatively slowly, at a rate of \(2.0\text{sec}^{-1}\), which results in a mean time to transition to the state fallingBelow of \(0.5\text{sec}\). When falling below the PONR the plunger falls faster at a rate of \(4.0\text{sec}^{-1}\). To keep the diagram uncluttered, and following the convention of the PRISM model checker [35], a default rate of one time unit\(^{-1}\) is assumed for transitions without an explicit rate. In our example, so that the internal workings of the plunger are relatively quick, we assume a base time unit of one tenth of a second, or decisecond (dsec). That is, the default rate of transitions in our model is \(10.0\text{sec}^{-1}\). We associate a rate (of \(1/10\text{sec}^{-1}\)) with the Operator module to capture the speed at which the button is successively be pushed or released.

Component faults. As with standard FMEA, a set of component failures modes is identified by the team of safety experts. However, in pFMEA the likelihood of those faults occurring is also determined. The behaviour of the system under a particular failure mode, the failure view, is specified using the same language as the probabilistic system model. The system then transitions from its working model to a failure view at the failure rate, \(\lambda\). If our model also allows for a recovery strategy, the system may transition from the failure view to the normal view with a recovery rate, \(\mu\).

There are several methods by which the failure view models can be constructed. The most straightforward is for each failure view to be constructed by-hand (working from the original) and saved individually. Alternatively, with adequate tool support, all failure views could be combined into the one model. This has lower overhead in terms of construction time than the first method, since multiple points of failure can be automatically and exhaustively generated. Such a model is more complex to construct, however, and its size may have implications for model checking. Finally, it may be possible to auto-generate failure views for particular classes of faults. The approach taken will depend on the tool support available for the modelling language being used.

Example. Consider the failure of the TopSensor component where it becomes stuck high and therefore never reports that the Plunger has left the top position. The normal behaviour of the TopSensor is shown in Figure 6 on the left. Initially the module is in state 0 awaiting the input atTop? from the Plunger. When this is received a message is output to the Controller indicating the plunger’s new position. When the Plunger has left the top position, the message notAtTop? is received and the Controller is notified by sending tsLow!.

In failure view mode the TopSensor will not send the message tsLow! to the Controller. The transition from state 3 to 0 becomes guarded: if in normal mode \(fv = 0\) then the message is sent as usual, else if in failure mode \(fv = 1\) the message is not sent. We use the notation guard/action in Figure 6 to model this guarded behaviour, i.e., \(fv = 0/\text{tsLow}!\) and \(fv = 1/\text{skip}\), respectively.

Note that the TopSensor with injected failure covers the behaviour for both normal and failure mode. This allows switching between the failure modes during runtime, and especially for a recovery of the fault.
HazardCondition

In pFMEA, hazards are inputs as in standard FMEA. Each hazard is associated with a tolerable hazard rate (THR), which constitutes the safety requirement the system must satisfy. The hazard condition may be formalised in a probabilistic temporal logic such as Continuous Stochastic Logic (CSL) [2] or Probabilistic CTL (PCTL) [25]. In CSL, for example, the following template for safety requirements could be used:

$$ P_{<T_{HP}} \left[ true \leq T \left( HazardCondition \right) \right] $$

In this formula, $T_{HP}$ is the tolerable hazard probability and $T$ is the observation time of the fault-injection experiment. It succeeds if the likelihood of HazardCondition occurring within $T$ time units is less than $T_{HP}$. The observation time $T$ is normally based on the mission time of the investigated system. The tolerable hazard probability ($T_{HP}$) for exponential distribution can be determined from the observation time and THR as follows [5]:

$$ T_{HP}(T) = 1 - e^{-T \times THR \times T} \tag{1} $$

The HazardCondition is normally a real-time temporal logical formula or a normal temporal logic formula, and may be formalised using specification patterns as given in [6, 18, 33].

Example▷ A potential hazard of the metal press system is that the motor is turned on while the plunger is falling fast. After risk analysis this hazard is associated with a tolerable hazard rate of $10^{-7}$ per hour [43], which is equivalent to Safety Integrity Level (SIL) 3 [9, 32]. For our model the hazard condition can be expressed via the state of the Plunger and Motor: if the Plunger is in state fallingBelow and the Motor in state switchOn then both modules are ready to synchronise on the event mOn? (see Figures 4 and 5). This behavior may lead to an accident because the kinetic energy of the plunger is too high and consequently the motor may burst and injure the operator. The complete safety requirement can be expressed in CSL as the following formula.

$$ P_{<T_{HP}} \left[ true \leq T \left( stateP = fallingBelow \land stateM = switchOn \right) \right] \tag{2} $$

To get a model checkable formula for the safety requirement (2) we set $T$ to the mission time of one year, which must be then translated to the base time units (i.e., $365 \times 24 \times 60 \times 60 \times 10$ if deciseconds are used as the unit of time). Based on equation (1) the $T_{HP}$ is 0.000876 for an observation time of one year. Consequently the final instantiated CSL formula is the following safety requirement 1 (cf., Table 2):

$$ 
\begin{align*}
\lambda_1 = \lambda_2 = \lambda_3 &= \frac{1}{(5 \times 365 \times 24 \times 60 \times 60 \times 10) \text{dsec}} \\
\text{Table 1. Failure view matrix for a double failure experiment} \\

<table>
<thead>
<tr>
<th>F_{v_0}</th>
<th>F_{v_1}</th>
<th>F_{v_2}</th>
<th>F_{v_{1,2}}</th>
<th>F_{v_{1,1}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\lambda_1$</td>
<td>$\lambda_2$</td>
<td>0</td>
<td>$\mu_1$</td>
</tr>
<tr>
<td>$\mu_1$</td>
<td>0</td>
<td>0</td>
<td>$\lambda_2$</td>
<td>0</td>
</tr>
<tr>
<td>$\mu_2$</td>
<td>0</td>
<td>0</td>
<td>$\lambda_1$</td>
<td>0</td>
</tr>
<tr>
<td>$\mu_3$</td>
<td>$\mu_2$</td>
<td>$\mu_1$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

With a failure rate of $\lambda_1$ the system can transition from normal mode to failure view $F_{v_1}$ (TopSensor stuck high) and with failure rate $\lambda_2$ from normal mode to $F_{v_2}$ (BottomSensor stuck high). With a rate of $\lambda_2$ the system can transition into the double failure view $F_{v_{1,2}}$ if it is already in mode $F_{v_1}$, and similarly with a rate of $\lambda_1$ can transition from $F_{v_2}$ to $F_{v_{1,1}}$.

From failure view $F_{v_1}$ the system can recover with recovery rate $\mu_1$, and from $F_{v_2}$ with $\mu_2$. In double failure view $F_{v_{1,2}}$ the recovery rate is $\mu_3$ to transition back to normal operation mode $F_{v_0}$.

In our fault-injection experiment for the press system the failure rates are equal for all Sensors:

$$ 
\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{(5 \times 365 \times 24 \times 60 \times 60 \times 10) \text{dsec}} 
$$

\begin{figure}
\centering
\includegraphics[width=\textwidth]{example.png}
\caption{The TopSensor module without (left) and with (right) a failure injected}
\end{figure}
Another hazard can occur when the motor is switched off while the plunger is rising. This may be formalised as follows (with the same observation time $T$ as above):

$$
\mathcal{P}_{<0.000876} [ \text{true} \mathcal{U}^{\leq 315560000} \left( \text{state}_P = \text{fallingBelow} \land \text{state}_M = \text{switchOn} \right) ]
$$

This safety requirement is also related to a tolerable hazard rate of $10^{-7}$ per hour (SIL 3) [43] and consequently this formula can be instantiated similarly to safety requirement 1 as formalised in (3). The resulting instance becomes our safety requirement 2 (cf., Table 2):

$$
\mathcal{P}_{<0.000876} [ \text{true} \mathcal{U}^{\leq 315560000} \left( \text{state}_P = \text{risingBelow} \lor \text{state}_P = \text{risingAbove} \land \text{state}_M = \text{switchOff} \right) ]
$$

Typically the result of an experiment will be a yes or no answer: either the system fulfils the safety requirement or not. If all safety requirements are satisfied, the current system design (including it failure and recovery rates) is acceptable. If any requirement is not satisfied then the current system design must be revised and improved to reduce the likelihood of the hazard occurring.

### 3.2 Experimental results using the probabilistic model checking tool PRISM

Based on the description of the pFMEA approach we have done several fault-injection experiments with the probabilistic model checker PRISM [35]. These experiments would normally be performed by safety experts during the FMEA process to get insight into the cause-consequence relations between failure modes and system hazards. There are two interesting classes of results: a failed experiment (safety requirement violated) and successful experiment (safety requirement not violated). The latter may be further subdivided into two cases: when the probability of the hazard occurring is zero and non-zero. In the zero case, there is no causal relationship between the injected failure mode and the hazard under investigation, and consequently it can be added to the FMEA table that the failure mode will not cause this hazard. In the non-zero case, this indicates that the hazard can occur but within the tolerable limit – however, if failure rates or THR are to change, the analyst may have to recheck the results. The FMEA table could then indicate that the failure causes the hazard only under unlikely conditions.

#### Example

To create the fault injection experiments for our metal press case study, we have used a straightforward PRISM-encoding of the system model and its failure views. To encode the failure view matrix, a new module, FailureViewManager, is specified, which represents the matrix as a CTMC. Each of the transitions of the CTMC has a specific transition rate and can be encoded similarly to the following example, which describes a transition of the system from correct operation (view=0) to failure view 1 (view’=1) with a transition rate characterised by the variable FailureR:

$$
[\text{view}=0 \rightarrow \text{FailureR:(view’}=1)\]
$$

The outcomes of some fault-injection experiments for interesting cases of the press system and its safety requirements are described in Table 2. This table shows that the failure mode, TopSensor stuck high, violates safety requirement 2 (as formalised in (5)), that the motor should not be switched off when the plunger is rising. In the third fault-injection experiment, BottomSensor stuck high, the system model violates the first safety requirement. All other fault injection experiments indicate no or not significant influence of the injected faults on the hazard.

The probabilistic model of each fault-injection experiment can also be used to derive recommendations for design improvements in order to reduce the probabilities of certain hazards. As an example, using the PRISM tool it is possible to perform a sensitivity analysis of the model with respect to specific parameters, like the failure rates or environment parameters. The results of such a sensitivity analysis for the TopSensor-stuck-high experiment and safety requirement 2 is presented in figure 7. Based on this graph, a safety analyst may, for example, recommend replacing the current TopSensor by a similar product with a higher MTTF. This recommendation would be added to the FMEA table.

Furthermore, PRISM includes facilities for calculating probabilities, in addition to providing yes/no answers for particular formulas. As a result, this gives a safety expert an indication how far the probability of the hazard condition lies above or below the threshold for the THP, and thus can help to adjust the repair measures, etc., to be taken.

One limitation of current probabilistic model checking tools, including PRISM, is the inability to produce counter examples. This is because there is no single trace which violates a probabilistic safety property, since some traces are expected to satisfy the property and some are expected to fail (except in the cases where the expected probability is 0.0 or 1.0). However, current research is addressing this issue by using tree-like counter examples (e.g. [24]).
### Table 2. Results of Selected Failure Injection Experiment

<table>
<thead>
<tr>
<th>Experiment Number</th>
<th>Description</th>
<th>Violation of Safety Requirement 1</th>
<th>Probability of Hazard 1</th>
<th>Violation of Safety Requirement 2</th>
<th>Probability of Hazard 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Normal Press Behaviour, No faults are injected</td>
<td>no</td>
<td>0.0</td>
<td>no</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>TopSensor stuck high, MTTF=5 years</td>
<td>no</td>
<td>0.0</td>
<td>yes</td>
<td>0.00226</td>
</tr>
<tr>
<td>3</td>
<td>BottomSensor stuck high, MTTF=5 years</td>
<td>yes</td>
<td>0.00207</td>
<td>no</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>TopSensor stuck low, MTTF=5 years</td>
<td>no</td>
<td>0.0</td>
<td>no</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>BottomSensor stuck low, MTTF=5 years</td>
<td>no</td>
<td>0.0</td>
<td>no</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>BottomSensor stuck high and TopSensor stuck high with the failure transition matrix and transition properties as given in Table 1</td>
<td>no (not significant)</td>
<td>0.0000170</td>
<td>no (not significant)</td>
<td>0.000000821</td>
</tr>
</tbody>
</table>

**Figure 7. Influence of the TopSensor’s MTTF on the Probability of the Hazard 2**

### 4. Related Work

Many approaches have been developed to support the FMEA process with standard, non-probabilistic model checking techniques. These approaches can be distinguished by their modelling language and tools used to perform the fault-injection experiments. As an example, Heimdahl et al. [27] use RSML-ε as the language to represent the system model. To perform Software Deviation Analysis (SDA) [45], which is basically a FMEA, the system model including its injected faults is translated into NuSMV code [12]. Cichocki and Górski [10, 11] describe an approach which utilises CSP [28] as a system specification language and the refinement checker FDR [19] is used to perform fault-injection experiments. In the ESACS (Enhanced Safety Assessment for Complex Systems) project [7] the system is modelled directly in NuSMV code and the fault-injection experiments are supported by the FSAP/NuSMV-SA platform [8]. An approach that explicitly models the system with the graphical Behavior Tree language [16], is presented in [22]. It utilises the model checking environment provided by the SAL tool suite [14]. This language has been recently extended to incorporate timed and probabilistic behaviour [23, 13]. Thus, this notation can be used as a graphical frontend to support the input to the probabilistic FMEA process, namely the model and its failure views.

Despite the fact that there are several model checking based approaches that try to automate the FMEA process, none directly address automating the calculation of probabilistic relationships between component failures and hazards. This omission is significant because safety requirements are specified as a combination of a hazard condition and a tolerable hazard rate/probability, and the component failures occur with a certain probability in the real system. The cause-consequence relationships identified by a standard model checker may be unlikely or even infeasible, yet there is no guidance for the safety analyst as to the extent of the problem. Consequently, the use of probabilistic model checking, as promoted in this paper, has significant advantages over the approaches that use standard model checking.

In contrast to safety evaluation and FMEA, probabilistic model checking is widely used for modelling dependable systems and analysing their quality early in the development process. As an example, approaches for performance evaluation are presented in [3, 4, 37, 38]. These approaches represent performance requirements as probabilistic logic formulae and utilise probabilistic model checking to explore if the system meets its performance requirements. A similar approach is also used for reliability and availability [36, 46] and performability evaluation [26]. Although these approaches operate in a different dependability domain, the probabilistic models developed for the different analyses can potentially be shared with the safety analysis approach of this paper.

### 5. Conclusion and Future Work

In this paper we have described a methodology for system safety analysis which formally captures and analyses probabilistic system behaviour. This work complements existing research which uses (standard) model checking to automate the FMEA process by formalising the calculation of hazard violations. The method may be used with any sufficiently powerful probabilistic modelling language which has tool support for model checking; in this paper we have used continuous time Markov chains and the model checker...
PRISM [35], using an industrial metal press system as an example. The method improves on existing automated support from FMEA because the analyst can include tolerable hazard rates in the calculations, and experiment with different component failure rates, factors which are crucial for safety analysis.

There are several aspects of the methodology which may be extended to make the process more systematic. There have been languages developed for expressing safety properties in a systematic way that are both readable as natural language statements and also translatable into a formal specification language. This extension could be similar to specific patterns as proposed for standard [18] and real-time temporal logics [33]. Such frameworks should be extended to incorporate tolerable hazard probabilities. An important aspect of the methodology is the generation of models that represent faulty behaviour of some of the components. For some classes of faults, it may be possible to automatically generate the faulty model from the working model given the component in question and the type of failure. For instance, the TopSensor-stuck-high failure, as used in the paper, is an example of an omission failure. Commission failures are those where an incorrect signal or message is sent. Injecting a commission or omission failure into a sensor model can potentially be automated since the changes are straightforward.

Acknowledgements: This work was produced with the assistance of funding from the Australian Research Council (ARC) under the ARC Centers of Excellence program within the ARC Center of Complex Systems (ACCS).

References


