

Semi-Automatic FMEA Supporting Complex Systems with Combinations and Sequences of Failures

Martin Walker, Yiannis Papadopoulos and David Parker
University of Hull, UK

Henrik Lönn
Volvo Technology Corporation, Sweden

Martin Törngren and DeJui Chen
Royal Institute of Technology, Sweden

Rolf Johansson
Mentor Graphics Corporation, Sweden

Anders Sandberg
Mecel AB, Sweden

Copyright © 2009 SAE International

ABSTRACT

Failure Modes and Effects Analysis (FMEA) is a well established safety analysis technique used for the assessment of safety critical engineering systems in the automotive industry. Although FMEA has been shown to be useful, the analysis is typically restricted to the effects of single component failures; even partial analysis of combinations or sequences of multiple failures is in practice considered too complex, laborious and costly to perform. In this paper, we describe a new technique in which FMEAs are semi-automatically built from the topology of a system and component-level specifications of failure data. The proposed technique allows an extended form of "combinatorial & sequential FMEA" in which assessment of the effects of combinations and sequences of failures becomes feasible and cost effective. We show how this technique can address difficulties encountered in classical FMEA and, drawing from a simplified brake-by-wire example, we show how it can improve the assessment of safety critical automotive systems.

1 INTRODUCTION

Failure Modes and Effects Analysis (FMEA) is an inductive technique in which analysts study a system to identify the potential component failure modes and then infer the effects of those failure modes on the rest of the system. System models, such as engineering diagrams or functional block diagrams, are typically used to assist the analysts in understanding how component failures can affect other parts of the system. Once complete, the analysts can then assess the likelihood of those failures and the severity of their effects before estimating the potential risk they pose. The system can then be redesigned to avoid or minimise the effects of the most critical failures identified in the analysis and thereby improve the safety of the system.

Although FMEA is generally seen as a useful technique, it remains manual, laborious and expensive to apply. The analysis typically remains incomplete, being restricted to the assessment of the effects of single component failures, as exhaustive assessment of combinations of failures is very difficult to achieve in large systems. If, for

example, a system has 1000 potential component failures, assessment of the effects of combinations of 2 such failures requires that the analysis is repeated approximately half a million times. This combinatorial explosion problem is further exacerbated in cases where the effects of component failures depend on the temporal ordering – or sequence – of those failures. This type of temporal safety analysis is simply impossible in a classical FMEA.

The problem of combinatorial explosion can be partially mitigated by automating the process, and a number of software tools have been developed to help with the process of conducting an FMEA. Unfortunately, they were often designed to assist solely with the clerical aspects of the analysis and enforce consistency. Therefore, the analysis of the system itself often remains a manual process – a team of analysts still has to study the system models, identify component failures, and determine their effects on the rest of the system. With the increasing complexity of systems, many of which integrate different technology domains, it is becoming more difficult to be able to perform thorough analyses and determine the full consequences of individual failure modes; a problem exacerbated by a trend towards shorter design periods. Even if the FMEA is restricted to single points of failure, as system complexity increases, it becomes more likely that the analysis will contain either errors or omissions.

To overcome this problem, there has been a considerable body of work in recent years focused on ways of automating FMEA. One such methodology makes use of fault simulation [1-5]. Simulation of this sort requires domain modelling which restricts its usage to domains for which simulators and their requisite models have been developed. For example, fault simulation has been successfully applied to electronic systems in [6]. However, such approaches are computationally very expensive and do not help to overcome the inability of FMEA to analyse combinations or sequences of failures.

Another way to master the complexity is to take advantage of the hierarchical description of a system. In [14] the authors propose a method where sub-components are analysed independently of the overall system, and the analysis that takes place on higher integration levels abstracts away from details that are refined on lower component levels. Although this strategy would improve the situation, combinatorial explosion in a hierarchical FMEA would still cause difficulties at each level of the analysis.

In this paper, we outline an approach to synthesis of combinatorial FMEAs using a recently proposed but by now well established compositional safety analysis technique called Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) [7]. This is a semi-automated approach to FMEA that can be applied to a greater range of systems, be they mechanical, electrical, hydraulic, software-based, or some

combination thereof. Because HiP-HOPS generates the FMEA via automatically generated fault trees – which are deductively produced, starting with the system failure and working back to determine the root causes – it does not suffer in the same way from combinatorial explosion when considering multiple failure modes. Furthermore, by making use of recent advances in temporal safety analysis – known as Pandora [8] – HiP-HOPS enables the analysis to incorporate the effects of different sequences of failures as well, resulting in a new form of 'sequential' FMEA.

In section 2, we describe recent extensions to the HiP-HOPS tool that enable semi-automatic synthesis of combinatorial FMEAs. In section 3, we explain how the technique is capable of producing sequential FMEAs by introducing Pandora, and then in section 4 we present a small case study that demonstrates the benefits of such an approach in producing more comprehensive and inclusive information about the safety of a system. Finally, in section 5 we present our conclusions and state how this new technique provides useful advances in the field of FMEAs.

2 COMBINATORIAL FMEA IN HIPHOPS

The HiP-HOPS tool performs its analysis on the basis of an annotated system model that provides the tool with the topology of the system: the components and the connections between them. The system model can be flat or hierarchical, with components containing architectures of subcomponents which can in turn contain more subcomponents. This also means that different models can be used at different stages of the design process; for example, a relatively simple functional block diagram might be used early in the design and could be refined with a schematic showing the detailed hardware architecture of the system. Later on in the design lifecycle, programmable components in a hardware architecture may be further refined, enclosing software architectures described using functional or data flow diagrams. In the context of work with EAST-ADL [15], it has also been shown that HiP-HOPS studies can also be performed on layered architectures that combine different levels abstraction, functional, hardware and software.

The annotations in the model provide HiP-HOPS with *local failure data*. This mainly takes the form of a set of logical expressions that show how failures at the component's outputs (known as output deviations) can be caused by a combination of input deviations – failures detected at the component's input (e.g. an omission of signal) – and internal failure modes of the component itself. Boolean operators such as AND and OR can be used to create the expression and this enables HiP-HOPS to analyse the effects of more than one failure mode occurring in combination.

Input and output deviations are given a *failure class*. This gives some indication of the nature of the failure and generally falls into one of three categories: provision, e.g.

an omission or unexpected commission of a signal; timing, e.g. late or early reception of an input; and value, which indicates an error in input/output (e.g. high pressure, low voltage etc). Internal failure modes do not have a failure class but can include failures of many different types, such as environmental influences external to the component (e.g. EMI, temperature), material failures of the component itself, and human error in the control of the component.

As an example of local failure data, consider a simple pump. It could respond to and originate failures in a number of ways, e.g.

- Omission-outflow = Omission-power OR mechanical_failure OR blockage OR Omission-inflow
- Commission-outflow = Commission-power
- Low-outflow = partialBlockage OR Low-power
- High-outflow = High-power

In some cases, such as with the commission of output flow, it effectively propagates a failure at an input to a failure at its output; in other cases, such as with the omission of output flow, there are many possible causes, both internal and external to the component. The failure class of a deviation is the first part, prior to the dash, and the input/output ("port") of the component is the second part of the deviation; internal failures such as "blockage" do not have a specific port as they originate within the component and do not represent effects on inputs or outputs.

Similar components can reuse the same local failure data. Because the data is mostly context independent, identical or similar failure data could be used for all pumps in the system, and similarly for other types of H/W or S/W components. This makes it easier to annotate large models by minimising the amount of manual work required; and annotations can even be stored as part of a component type in a library so that the local failure data is already present when the component is added to the model¹.

Since the effects of component failures are rarely limited to the components themselves, a key objective in safety analysis is tracing the propagation of component failure through the model up to a related hazard on system level. HiP-HOPS achieves this by linking the output deviations of one component to the input deviations of another, so for example an omission of flow from a pump could cause a hydraulic actuator to stop functioning which in turn could have wider ramifications on the rest of the system. HiP-HOPS even allows finer control over this propagation by also allowing the connections between components to be annotated with logical expressions.

¹ These ideas for reuse of component failure data have been implemented in a commercial tool called Simulation X, which has recently been extended with an interface to HiP-HOPS.

HiP-HOPS starts this process at the outputs of the system as a whole by taking a system failure (a hazard when doing safety analysis) and then working backwards through the model, following the connections between components, until it can go no further. The result of this is a set of interconnected fault trees which show how individual component failures in the model can combine to cause one or more system failures.

These fault trees can then be flattened using qualitative fault tree analysis (FTA) techniques to obtain the *minimal cut sets* of the fault tree. These are the combinations made up of the smallest numbers of failures necessary to cause a system failure, and illustrate the direct links between original cause and ultimate failure. It is then possible to use these results to construct a multiple-event FMEA, showing not only the direct effects on the system of each failure mode, but also the further effects that failure mode can have if it occurs in conjunction with other failure modes (i.e. the other failure modes in the minimal cut sets).

For example, assume that HiP-HOPS-style FTA produced five minimal cut sets for two system failures as follows:

- System Failure 1 = {A}, {B, C}
- System Failure 2 = {A}, {B}, {C, D}

The cause-effect relations captured by the HiP-HOPS tool for the conditions specified above is shown in the simplified FMEA Table 1 below²:

Failure	Further Failure	Effect
A	None	Sys Failure 1
		Sys Failure 2
B	None	Sys Failure 2
	C	Sys Failure 1
C	B	Sys Failure 1
	D	Sys Failure 2
D	C	Sys Failure 2

Table 1: Combinatorial FMEA

The table illustrates several features that would not be possible in a normal, single-failure FMEA. For example, it shows that although B can cause System Failure 2 by itself, it can also cause System Failure 1 in conjunction with C. It also shows that both C and D can cause system failures if they occur with one other event (including each other); in an ordinary FMEA, neither C nor D would be shown with any effects since they have no direct effect on the system. The table also shows that A is a common cause of two system failures – a clear

² A typical FMEA, and indeed the FMEAs generated by the HiP-HOPS tool, contain more columns which for simplicity and clarity have been omitted in Table 1. Only the important relationship between causes (component failures A, B, C) and effects (system failures 1 and 2) is explored here

vulnerability in the design – but this is a result which an ordinary FMEA should also produce.

Furthermore, because this table was generated from a fault tree analysis – a deductive method, working backwards from the system failure to determine the root causes – it avoids much of the combinatorial explosion that afflicts inductive techniques like FMEA, where the analysis starts with the failure and attempts to determine its effects. This in practice means that, once a system model has been annotated with local failure/propagation data, the whole process would typically take only seconds to perform on systems that contain hundreds of components.

For the same reason, HiP-HOPS also makes it possible to perform a quantitative analysis on the system, determining the probability of system failures by calculating the probabilities of individual failure modes, given that proper input failure frequencies are estimated. There are a number of formulae available, including simple constant failure & repair rates or more complex Weibull failure models. This enables the analyst to see what contribution each failure mode makes towards the system failure it ultimately causes, and efforts to improve the reliability of the system can be focused accordingly. This kind of quantitative analysis is however only applicable for the H/W sub-systems, especially when analysing safety properties.

Finally, system failures can also be assigned severity or criticality values to give an indication of the seriousness of their consequences, thus generating a so-called FMECA (Failure Mode Effect and Criticality Analysis). In combination with the probability values, HiP-HOPS can then calculate a risk value for each system H/W failure. This adds extra information to the combinatorial FMEA and makes it possible to compare unlikely but catastrophic events with more likely but less critical failures.

All of these values are calculated automatically from the component annotations; the more information provided, the more comprehensive the results. Overall, the result is a much more powerful and expressive form of FMEA that provides the analyst with a greater degree of understanding of the system failure behaviour, therefore making it easier to decide which parts of the system need to be improved to increase reliability and safety.

3 INTRODUCING SEQUENCES TO FMEA

HiP-HOPS allows us to create FMEAs that take into account the effect of multiple failure events occurring in combination, but what if the sequence in which they occur is also important? This section describes how HiP-HOPS has been extended to be able to analyse this sort of dynamic behaviour, which is increasingly important in complex systems.

For example, consider the simple system in Figure 1, consisting of two components – a primary and a standby

– and a switch that activates the standby when the primary fails.

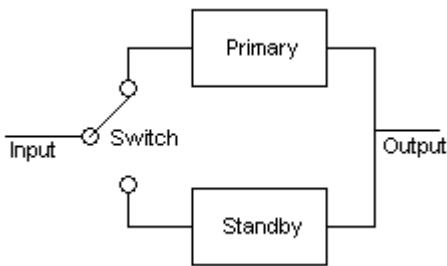


Fig 1: Simple system with dynamic failure behaviour

This behaviour should allow the system to continue operating even if one component fails. If we were to analyse this system using classical FMEA techniques, we might produce results such as those in Table 1:

Component	Failure	Effect
Primary	ComponentFault	Causes stand-by to activate
Switch	SwitchFault	Standby will not activate
Standby	ComponentFault	System failure

Table 2: Simple FMEA

However, these results do not truly represent the failure behaviour of the system. Although it indicates that a failure of the switch will mean the standby does not activate, it gives no indication of the sequence of events – and in this case, the sequence of events can make a large difference. If the switch fails *after* the primary has failed, then it has no effect on the system because the standby has already been activated; whereas if it fails first, before the primary fails, then it will lead to a system failure because the standby will not be activated. Thus the results in Table 1 are incomplete or at best pessimistic.

It would be better if we had some way of representing the sequence of events to better model the dynamic behaviour of the system. One approach is the Dynamic Fault Tree methodology [9, 10]. DFTs use a set of special logic gates such as FDEP (functional dependency) and SEQ (sequence) to describe the dynamic behaviour of a system, and a probabilistic analysis is then carried out using Markov chains. However, the DFT methodology is designed for quantitative fault tree analysis rather than the primarily qualitative analysis approach used in an FMEA, which means that DFTs are not the best solution to this problem. Another option is to represent sequences of events as events themselves – a kind of conditional event. For example, to represent the situation where the switch fails first, we could define an event such as "SwitchFailure AND PrimaryFailure AND SwitchFailure