A Method and Tool Support for Model-based Semi-automated Failure Modes and Effects Analysis of Engineering Designs

Yiannis Papadopoulos, David Parker
Department of Computer Science
University of Hull
HU6 7RX, U.K.
{y.i.papadopoulos,d.j.parker}@hull.ac.uk

Christian Grante
Volvo Car Corporation
Götenborg
SE-40531, Sweden
cgrante@volvocars.com

Abstract

Limitations in scope but also difficulties with the efficiency and scalability of present algorithms seem to have so far limited the industrial uptake of existing automated FMEA technology. In this paper, we describe a new tool for the automatic synthesis of FMEAs which builds upon our earlier work on fault tree synthesis. The tool constructs FMEAs from engineering diagrams (e.g. developed in Matlab-Simulink) that have been augmented with information about component failures. To generate a system FMEA, the tool first generates a "forest" of interconnected system fault trees by traversing the system model. This "forest" is then mechanically translated into a simple table of direct relationships between component and system failures, effectively a system FMEA. We describe the architecture of the tool and demonstrate its application on a steer-by-wire prototype. We also discuss its performance and show that this approach could lead to efficient ways of generating useful analyses from design representations.

Keywords: model-based FMEA, fault tree synthesis, steer-by-wire systems, automated safety analysis.

1 Introduction and background

Failure Modes and Effects Analysis (FMEA) is a classical system safety analysis technique which is currently widely used in the automotive, aerospace and other safety critical industries. In the process of an FMEA, analysts compile lists of component failure modes and try to infer the effects of those failure modes on the system. System models, typically simple engineering diagrams, assist analysts in understanding how the local effects of component failures propagate through complex architectures and ultimately cause hazardous effects at system level.

Although there is software available that assists engineers in performing clerical tasks, such as forming tables and filling in data, the intelligent part of an FMEA process remains a manual and laborious process. Thus, one of the main criticisms of FMEA (Hawkins *et al.*, 1996) is that the time taken to perform the analysis can often exceed the period of the design and development phases and therefore the analysis *de facto* becomes a mere deliverable to the customer and not a useful tool capable of improving the design. Difficulties naturally become more acute as systems grow in scale and complexity.

To address those difficulties, a body of work is looking into the automation and simplification of FMEA. To mechanically infer the effects of component failures in a system, several approaches have been proposed which use quantitative or qualitative fault simulation. The reader is referred, for example, to the work of Lehtela (1990), Bull and Burrows (1996), Price and Taylor (2002) and Xu *et al. (*2002). The application of these approaches have been demonstrated successfully mainly in the domains of electronic and electrical circuits. Despite substantial progress in the development of this technology, though, fault simulation requires domain modelling and is therefore restricted to domains for which models and simulators have been developed to facilitate the generation of an FMEA. Thus, limitations in scope but also difficulties with the efficiency and scalability of algorithms seem to have so far limited the industrial uptake of automated FMEA technology. The problem therefore persists and hence there is scope for new approaches to the synthesis of FMEAs. One of the challenges for research is to find a generic solution, i.e. one that is applicable on (so called) mechatronic systems that bring together diverse technologies, i.e. mechanical, hydraulic, electrical, and programmable electronic.

In this paper, we propose a new approach to the synthesis of FMEAs which builds upon recent work towards automating fault tree analysis (Papadopoulos *et al.*, 2001). In this approach, FMEAs are built from engineering diagrams that have been augmented with information about component failures. The approach is only semi-automatic, i.e. some annotations must be added to the system model before an FMEA can be generated. However, the effort required to make these annotations is compensated by gains in terms of scope. Indeed, the proposed approach is generic, i.e. not restricted to an application domain, and potentially applicable to a range of widely used engineering models.

The proposed approach and the FMEA synthesis tool are extensions to an earlier tool for the automatic synthesis of fault trees described in Papadopoulos and Maruhn (2001). In section 2 of this paper we outline this approach, in section 3 we present the architecture of the tool and in section 4 we discuss an application of this tool in the

early stages of the design of an embedded vehicle control system.

2 Approach

In the proposed approach, FMEAs can be constructed from models developed at various stages of the design life-cycle. The models that provide the basis for the analysis should identify the topology of the system, i.e. the system components and the material, energy or data transactions among those components. Models can also be hierarchically structured and record in different layers the decomposition of subsystems into more basic components. We should note that this type of structural models include piping and instrumentation diagrams, functional block diagrams, data flow diagrams and other models commonly used in many areas of engineering design.

The first step in the analysis of such models is the establishment of the local failure behaviour of components in the model as a set of failure expressions which show how deviations of component outputs can be caused by internal malfunctions and deviations of component inputs. Input and output deviations referenced in these failure expressions are described qualitatively and typically represent extreme conditions such as the omission or commission of parameters or qualitative deviations from correct value (i.e. hi-low) and expected timing behaviour (i.e. early-late)[1]. Collectively, a set of failure expressions that logically explain all possible deviations at all output ports of a component provides a model of the failure behaviour of the component under examination. This model can be developed once and then stored in a library. For simple components, e.g. sensors and actuators, such models could be re-used across different applications to simplify the manual part of the analysis and the overall application of the proposed technique.

Once these local failure analyses have been inserted in the system model, the structure of the model is then used to automatically determine how local failures propagate through connections in the model and cause functional failures at the outputs of the system. This global view of failure is initially captured in a set of fault trees which are automatically constructed by traversing the model of the system backward moving from the final elements of the design, i.e. the actuators, towards system inputs and by evaluating the failure expressions of the components encountered during this traversal.

The fault trees synthesized using this approach show how functional failures or malfunctions at the outputs of the system are caused by logical combinations of component failures. These fault trees may share branches and basic events in which case they record common causes of failure, i.e. component failures that contribute to more than one system failures. Thus, in general, the result of the fault tree synthesis process is a network of interconnected fault trees which record logical relationships between component and system failures as this is illustrated in figure 1. The top events of these fault trees represent system failures. Leaf nodes represent component failure modes while the body of intermediate events (and intervening logic) records the propagation of failure in the system and the progressive transformation of component malfunctions to system failures. One difficulty here is that in large and complex systems fault trees tend to grow very large which means that the structure of the fault tree (i.e. intermediate events and logic) is typically too difficult to inspect and interpret meaningfully[2].

In the final step of the proposed process, this complex (and often impossible to interpret) body of fault propagation logic is removed from the analysis by an automated algorithm which translates the network of interconnected fault trees into a simple table of direct relationships between component and system failures. In a similar way to a classical FMEA, this table determines for each component in the system and for each failure mode of that component, the effect of the failure mode on the system. The table shows whether, and how, each failure mode causes one or more system failures (i.e. top events of fault trees) by itself or in conjunction with other events. The concept is illustrated in figure 2.

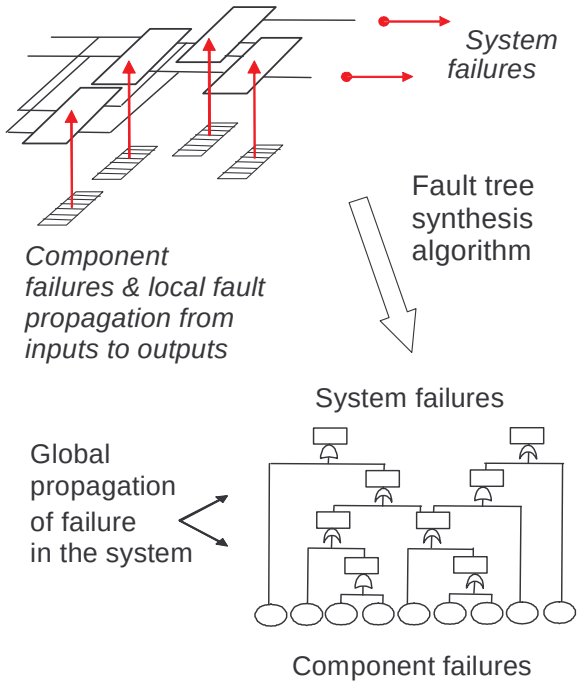
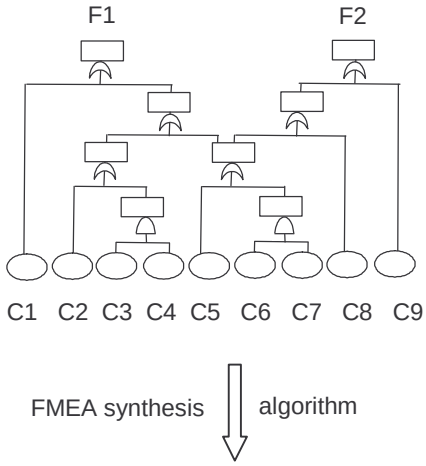


Figure 1: A network of automatically created fault trees

[1] For discussion of qualitative failure modes and their application in computer system and software hazard analysis, in particular, the reader is referred to McDermid *et al.* (1995) and Ministry of Defence, U.K. (2000).

[2] This is to a large extent true for all fault trees whether manually or automatically constructed. In manual fault tree analysis, though, experienced safety analysts may be able to use their knowledge in order to structure large fault trees in a more comprehensible and useful manner.

Network of Interconnected System Fault Trees



Multiple Failure System FMEA

Component failure	Direct effects on the system	Effects caused in conjunction with (other events)
C1	F1	-
C2	F1	-
C3	-	F1 (C4)
C4	-	F1 (C3)
C5	F1,F2	-
C6	-	F1,F2 (C7)
C7	-	F1,F2 (C6)
C8	F2	-
C9	F2	-

Figure 2: Synthesised fault trees and FMEA

Note that in a classical manual FMEA only the effects of single failures are typically assessed. Thus, one advantage of generating an FMEA from fault trees is that fault trees record the effects of combinations of component failures and this useful information can also be transferred into the FMEA.

To accommodate this additional information, the resultant FMEA tables are split into two, one containing the direct effects on the system, i.e. those effects caused by single component failures, and the other containing further effects, i.e. those effects caused by two or more component failure modes. This allows separate, easy access to the most critical information, the single points of failure. Perhaps more importantly, the FMEA shows all functional effects that a particular component failure mode causes. The latter is particularly useful as a failure mode that contributes to multiple system failures (e.g. C5 in the example of figure 2) is potentially more significant than those that only cause a single top event. Precisely because it records the effects of combinations of component failures, this type of FMEA can, in practice, help analysts not only to locate problems in the design, but also to determine the level of fault tolerance in the system, i.e. to determine whether the system can tolerate any single or any combination of two, three or more component failures.

3 Tool

To enable the practical and useful application of the above concept in engineering design, we have developed a tool that generates FMEAs from models developed in Matlab Simulink. Simulink was chosen as a modelling environment because it is both a widely used engineering tool and a tool for which in the past we have developed an automated fault tree synthesis algorithm. It should be noted, however, that the applicability of the proposed technique is not restricted to Simulink models. Any model that provides the topology of the system, i.e. components and connections, is suitable for this type of analysis. In Papadopoulos and Petersen (2003), for example, we have demonstrated synthesis of fault trees from models of marine system designs developed in Simulation X.

3.1 Architecture and Algorithms

Three major extensions to the original fault tree synthesis tool were made to enable the construction of FMEAs. Firstly, the fault tree synthesis algorithm was improved to allow the simultaneous synthesis of more than one interconnected system fault trees. In a single step of execution, the tool generates a complete set of system fault trees with top events that represent all possible deviations of system parameters at all outputs of a system. The second significant extension made was the addition of a minimal cut-set calculation algorithm. The cut-sets of the synthesised fault tree trees are calculated by applying a depth-first, bottom-up traversal strategy, in the course of which the logic of each tree is progressively established and then simplified using classical Boolean reduction techniques (Semanderes, 1971). The final significant addition to the original tool was that of an FMEA synthesis algorithm that processes all cut-sets to establish direct relationships between component failures and system failures in two FMEA tables (i.e. direct effects, and effects caused in conjunction with other events).

The architecture of the tool is illustrated in figure 3. The first component of the tool is a graphical user interface that analysts can use to annotate components with the failure annotations required for the fault tree synthesis. This data becomes part of the Simulink model and is automatically saved and retrieved by Matlab every time the model is opened or closed by a user. Failure annotations reference only attributes of the corresponding components (i.e. failure modes and deviations at component input or output ports). This means that such annotations can, in principle, be re-used within the same model or across different models with the obvious benefit of simplifying the manual part of the analysis.

Once a model has been annotated, it is saved by Matlab in the format of a Simulink model file. The second component of the FMEA tool is a parser that interprets such files, and reconstructs the enclosed annotated models for the purposes of fault tree synthesis. The synthesis itself is performed by the third component of the tool, the fault tree synthesis algorithm.

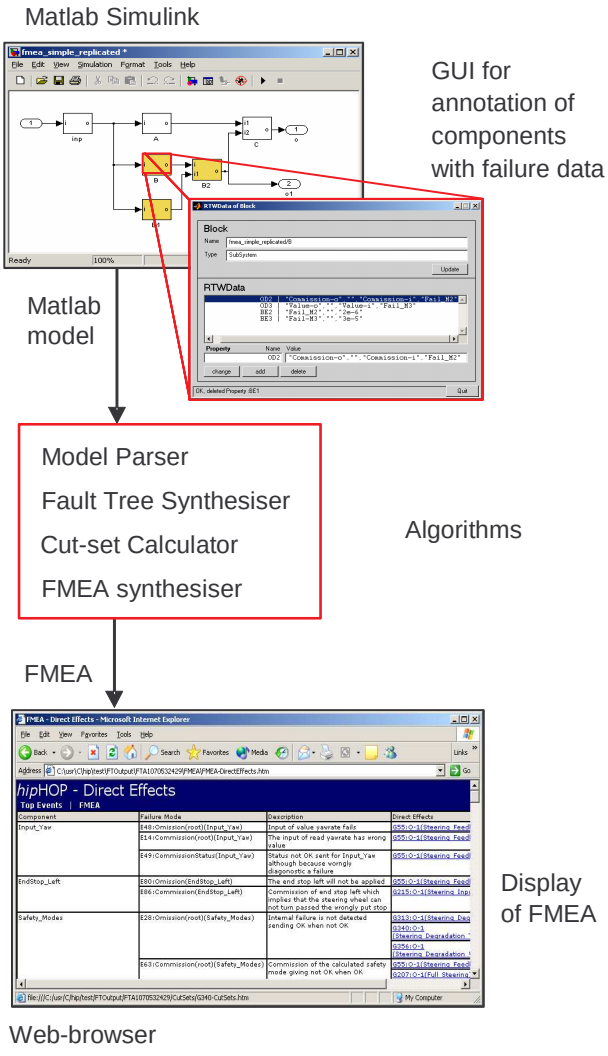


Figure 3: Architecture of the FMEA tool

To generate fault trees, the algorithm performs a backward traversal from each output of the model, in the course of which it evaluates the failure expressions contained in the local analyses of the components encountered during the traversal. The resultant network of fault trees is then logically reduced into minimal cut-sets. Finally, an FMEA synthesis algorithm operates on these cut-sets, and in a single traversal of the cut-sets generates the two FMEA tables. In the current implementation of the algorithm, the synthesis of the FMEA is separated from the display of tables. Indeed, an FMEA store is first created in memory and then an HTML generator is used to parse this store and create web pages containing the tables of data. The advantages of this medium include easy distribution and display and the ability, through hyperlinks, to navigate different aspects of the information.

3.2 Scope and performance

Substantial engineering work was performed to enable application of the general concept that we outlined above on realistic system models. The tool was designed, for example, to recognise and handle circular paths in the model that create circular references to the same failure logic in fault trees. When such circles (e.g. representing control loops) are encountered, the failure logic contained in the circle is only incorporated once in the trees. At the same time, a note is made by adding a special node in the trees which does not affect the rest of the failure logic but clearly warns about the earlier existence of circular logic at this point.

To deal with hierarchical models effectively, the synthesis algorithm was designed to perform traversals both across the vertical and horizontal axis of the design hierarchy. Indeed, the current implementation allows the annotation of hierarchical structures at all levels of the design. Failure analyses at sub-system level are important as they help to collectively capture the effect of failure conditions that do not necessarily require examination at basic component level. If, for example, a subsystem as a whole is susceptible to some environmental disturbance like electromagnetic interference, then the effects of this condition can be directly specified with a failure annotation at subsystem level. This annotation, for example, could define that all outputs of the subsystem are omitted in the event of electromagnetic interference. Such annotations would typically complement other annotations made at the level of the enclosed components to describe aspects of failure behaviour at this level (e.g. the mechanical and electrical failure modes of each component). In general, when examining the causes of a failure at an output of a sub-system, the fault tree synthesis algorithm creates a disjunction between any failure logic specified at sub-system level and logic arising from the enclosed lower levels. Thus, by enabling causes of failure to be described at both component and sub-system level, it becomes possible to avoid repetition of data that would otherwise be required to describe factors affecting entire sub-systems.

The FMEA tool is also designed to handle complications caused in the traversal of the model and the fault tree synthesis by the multiplexing and de-multiplexing of flows that often exist in Simulink models. It also recognises and handles relayed control signals (triggers) and the propagation of failure between components that communicate remotely using implicit protocols (Data-store/Data-read pairs, for example). Such features are essential to make this technique applicable on complex models and render the tool useful in industrial contexts of application. They have been achieved by incorporating into the tool syntactical and semantic information about Matlab-Simulink components. The tool, for example, can recognise “Data-read” components which are used in Simulink models to read data flows generated remotely by corresponding “Data-store” components. When such components are encountered during the fault tree synthesis, a global search is performed to locate the remote “Data stores” from which potential deviations of system parameters may originate.

The speed and performance of the FMEA tool are also crucial in achieving scalability and industrial applicability. We have not had a chance yet to perform a rigorous performance evaluation of the proposed algorithms. First applications indicate, though, that this approach can lead to fast and efficient ways of generating useful safety analyses from system design

representations. An indication of the present performance of the tool is that it is taking a little more than a minute in an average personal computer to generate an FMEA from an architectural model of a steer-by-wire system that contains more than a hundred components and generates several thousand cut sets (see also case study in section 4). This result refers to an FMEA that records the effects of up to four simultaneously occurring component failures modes. When this limit is set at two, the time dramatically decreases, obtaining timings in the order of a few seconds. To the best of our knowledge, these speeds compare favourably with other results reported in the literature of automated FMEA where systems have been reported to take hours even when considering only the effects of single component failures. Direct comparisons, however, are not possible because the proposed approach leads only to semi-automatic synthesis of FMEAs, while most other work aims to fully automate the process.

The speed of the tool currently seems sufficient to deal with relatively complex design problems in which annotated components are in the order of hundreds. Problems may arise, though, in large systems that contain thousands of annotated components and may result in a failure logic that is composed of millions of cut-sets[3]. The most susceptible part of the tool to problems of scale is the calculation of cut-sets, a computationally expensive operation, and a problem where a lot of contemporary research is focused. To further improve the speed of the tool in this area, we are currently considering using a recently proposed minimal cut-set calculation algorithm (Sinnamon and Andrews, 1997). The algorithm pre-processes fault trees, converting them into Binary Decision Diagrams, using ordering rules to determine the position of failure modes in the hierarchy of the tree. We hope that the improvements in efficiency that could be achieved by using this algorithm will further improve the scalability of the proposed techniques and ultimately enable their application in problems of industrial scale.

4 Case study

The method and tool are currently being evaluated by Volvo cars in a case study of medium complexity (i.e. hundreds of components) performed on a Matlab-Simulink model of an advanced steer-by-wire prototype system for cars. Note that classical FMEA is typically applied towards the end of the design life-cycle when details about the components of the system and their failure modes are available. Volvo, however, is using the tool in a rather different way, in order to drive the design of this system from the early stages of its development.

In this project, an FMEA was performed on an abstract functional model of the system. The objectives of the experiment were two-fold: first, to evaluate whether application of the technique could assist the early identification of potential design flaws; second, to evaluate whether the analysis of a functional model could point out critical functions and guide the design of such functions. Satisfaction of these two objectives would mean that:

- expensive design iterations needed to correct errors late in the design could be avoided
- an effective top-down design approach could be established in which the design of critical functions (and safety measures for these functions) could be driven by the result of FMEAs performed on increasingly more detailed models of the system

The functional model was developed in Matlab Simulink and was deliberately designed without any degraded or fallback modes, in order to test whether the results from the analysis could help in the systematic identification and design of such modes. The model identifies input, processing and actuator functions and how the interaction among these basic functions results in the provision of system functions such as the control of steering, the generation of driver feedback in the form of torque on the steering wheel, and the ability to send and receive information from and to controls in the steering wheel, e.g. the horn.

The model was annotated with information about the local behaviour of functions and then sixteen interconnected fault trees and an FMEA were automatically constructed by the safety analysis tool. Two qualitative failure modes were considered during the analysis, the omission and the commission of functions. Given the exploratory nature of the analysis, it was considered that it would be beneficial (for simplicity and ease) to aggregate value and timing failures into a single category of coarse commission failures. Omission failures were, therefore, used to represent the loss of functions, while the definition of commission failures was broadened to encompass conditions in which functions are provided in the wrong temporal context or at incorrect value. It could be argued that this reduction in the granularity of the analysis could have a negative impact on the quality of results. On the other hand, the abstract nature of the design model meant that detailed analysis of value and timing failures at this stage would have added very little value to the analysis, hence we opted for the simplification.

The resultant FMEA shows how omission and commission failures of input, processing and actuator functions in the model of the steer-by-wire system cause system level effects, i.e. omission or commission of steering, driver feedback and other functions. A classification of the severity of those effects into marginal and catastrophic helped to identify the criticality of causes, i.e. failures of input, processing and actuator functions, and this in turn provoked decisions about the design of these basic functions. For example, wherever the analysis indicated that the omission of a function had only marginal effects while the commission had catastrophic effects, a design recommendation was made to design the function in a way that it fails silent. This in turn led to the identification of several degraded modes in

[3] One way to simplify the analysis of complex hierarchical models, which at low levels incorporate thousands of components, is to perform the annotation at a higher level of abstraction where there are a smaller number of components or subsystems to annotate.

which non-critical steer-by-wire functions may fail silent with only marginal effects on the system. A hierarchical state-chart was then constructed to show how graceful transition to such modes could be achieved. Driven by these results, a design iteration is currently underway to incorporate new degraded modes in an improved version of the system model. A more detailed description of the case study is beyond the scope of this paper. However, to illustrate the useful application of the method, we discuss an example based on a small and manageable fragment of model. Figure 4 illustrates a simple standby-recovery system in which function P processes the value generated by input function I. When omission at the output of P is detected, a redundant function S is initiated to replace P. The following expressions describe the failure logic of functions I, P and S (note that this logic has been simplified for the purposes of the example, e.g. there is no analysis of commission failures).

I: Omission-pvalue = Failed(I)
P: Omission-output = Omission-input *or* Failed(P)
S: Omission-output = Omission-monitor *and* (Omission-input *or* Failed(S))

From the above expressions and the structure of the model, the safety analysis tool generates the fault trees and FMEA illustrated in figure 5. The FMEA shows that a failure of the input function I causes omission of both the normal and standby outputs of the system and is, therefore, a critical failure. On the other hand, a single failure of P causes only omission of normal output and can, therefore, be seen as less critical. Finally, failure of S does not have any direct effect on the system. It becomes significant only in conjunction with failure of P which in this design provides the condition that precisely triggers the need to deploy S in the first place.

The FMEA indicates that input function I is the critical element in this design, representing a hazardous dependency between the two redundant processing functions P and S. Failure of I is, indeed, a direct cause of a critical system failure (omission of the standby output) and should, therefore, be made unlikely by design. On the other hand, the analysis shows that an independent failure of either P or S cannot cause a critical system failure. Emphasis in the design should, therefore, be placed on how to protect these two functions from common cause failures such as those caused by electromagnetic interference.

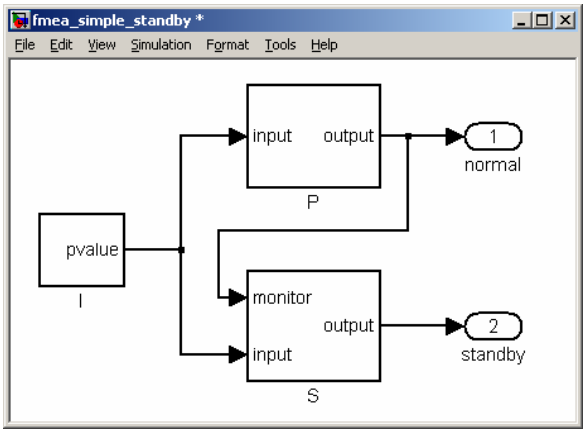
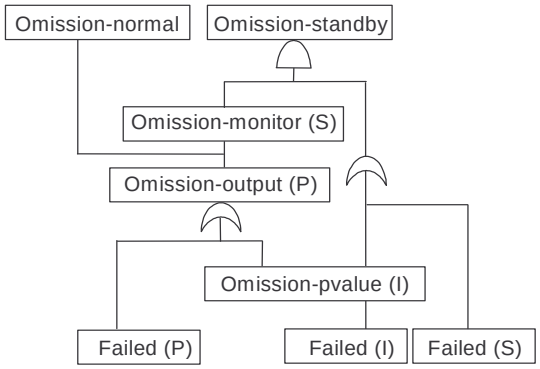


Figure 4: Example model

Network of System Fault Trees



Equivalent FMEA

Component	Failure mode	Direct effects on the system	Effects in conjunction with (other events)
I	Failed	Omission-normal Omission-standby	-
P	Failed	Omission-normal	Omission-standby (Failed(S))
S	Failed	-	Omission-standby (Failed(P))

SEVERITY OF EFFECTS ON THE SYSTEM:

Omission-normal	Marginal
Omission-standby	Critical

Figure 5: Fault trees and FMEA for the example

The above example demonstrates the ability of the synthesis tool to detect hazardous dependencies in the model, i.e. component failures that may cause simultaneous failure of hypothetically independent system functions. This may seem a trivial task in this example largely because the source of the dependency is very close to the affected functions but also because the model and associated failure logic are very simple. However, in reality, hazardous dependencies are not always as simple to detect especially those originating from remote energy and data sources which are deeply hidden in the hierarchy of complex designs. The detection of such dependencies is, indeed, a hard task which justifies, we believe, the provision of useful automated support to designers and analysts.

5 Conclusion

Safety analysis processes must evolve to deal with the difficulties posed by increasing complexity in technology and tighter integration of functions in the design of computer-based safety critical systems. Integrated safety-directed design processes need to emerge that can be effectively driven by the results of the assessment. In the context of such integration, classical safety analysis techniques like FMEA should ideally be automated (at least to some extent and without loss of effectiveness) to enable fast and cost effective iterations of system modelling and safety analysis that can meet the tight constraints of modern production.

In this paper, we proposed a concept and a tool for the model-based synthesis of FMEAs in which FMEAs are semi-automatically constructed from engineering design models that have been augmented with information about the local propagation of failures. In theory, the proposed process is largely automated and could be easily iterated to enable the continuous assessment of evolving designs. The potential benefits from application of this technique are substantial and include simplifying the analysis, easing the examination of effects of design modifications on safety and keeping the safety analyses consistent with the design.

There is not yet sufficient project experience to judge the practicability of this approach. However, early experience suggests that it is relatively easy to produce the required failure annotations, and that some re-use of this information is possible. The technique, therefore, is likely to give economic benefits. There are, however, many issues still to be addressed and many areas for potential improvement. For instance, to improve the value of the analysis, we need to obtain evidence that the local failure propagation specified by analysts, especially that of complex programmable modules, is no worse than assumed in the given failure annotations. The verification of such assumptions becomes naturally more important at the late stages of the design process when the design is finalised. Model checking and hazard directed testing are two methods that we currently consider as potential ways of verifying such assumptions.

There is also a broad issue of how to make safety a more controlled facet of the design so as to enable early detection of potential hazards and direct the design of preventative measures. The Volvo experience suggests that the proposed technique is applicable on abstract functional models and results could guide the refinement of abstract designs. However, the problem is largely unexplored, and further work is needed in this direction. In the context of our plans for future work, we currently explore ways for combining this work on model-based safety analysis with recent advances on evolutionary computation (Densig, 1997) (Papadopoulos and Grante, 2003) in order to achieve decision support in the allocation of safety and reliability requirements in the course of the evolution of a system design.

6 Acknowledgements

The authors would like to thank Volvo Cars Corporation for funding this work. In particular, we would like to thank Johan Wedlin and Mats Willanders for their comments and support in the development of these ideas.

7 References

Bull, D. R., Burrows, C. R., Edge, K. A., Hawkins P. G. and Woollons D. J. (1996): A tool for FMEA of hydraulic systems, IMECE '96, Int'l Mechanical Engineering Congress and Exposition, Atlanta, Georgia.

Denzig, B., F., Altiparmak and Smith A. E. (1997): Efficient Optimisation of All-Terminal Reliable Networks Using an Evolutionary Approach, IEEE transactions on Reliability, R**46**:18-26.

Hawkins, P. G., Atkinson, R. M., Woollons, D. J., Bull, D. R. and Burrows, R. (1996): An approach to FMEA using multiple models' IFMAA'96, Int'l Functional Modelling Application Association Conf., Athens.

Lehtela, M. (1990): Computer-Aided FMEA of Electronic Circuits, Microelectronics and Reliability, **30**(4):761-773.

McDermid, J.A., Nicholson, M., Pumfrey, D.J. and Fenelon, P., (1995): Experience with the application of HAZOP to computer-based systems, COMPASS '95:10th Annual Conference on Computer Assurance, pp. 37-48, Gaithersburg.

Ministry of Defence, U.K. (2000): Defence Standard 00-58: Hazard and Operability Studies on Systems Containing Programmable Electronics, Issue 2 dated 19 May 2000.

Papadopoulos, Y. and Grante, C. (2003): Techniques and tools for automated safety analysis & decision support for redundancy allocation in automotive systems, COMPSAC'03, 27th IEEE Int'l Conf. on Computer Software and Applications, pp. 105-110, Dallas, Texas.

Papadopoulos, Y., McDermid, J., Sasse, R. and Heiner G. (2001): Analysis and synthesis of the behaviour of complex programmable systems in conditions of failure, Reliability Engineering and System Safety, **71**:229-247.

Papadopoulos, Y. and Maruhn M. (2001): Model-based automated synthesis of fault trees from Matlab-Simulink models, DSN'01, Int'l Conf. on Dependable Systems and Networks, pp. 77-82, Götenborg.

Papadopoulos, Y., Petersen, U. (2003): Combining ship machinery system design and first principle safety analysis, IMDC'03, 8th Int'l Marine Design Conf., pp. **1**:415-426, Athens.

Price, C. J. and Taylor, N. (2002): Automated multiple failure FMEA, Reliability Engineering and System Safety, **76**:1-10.

Semanderes S. N. (1971): 'ELRAFT', a computer program for the efficient logic reduction analysis of fault trees, IEEE Transactions on Nuclear Science, NS-**18**:481-487.

Sinnamon, R. M., Andrews, J. D. (1997): New approaches to evaluating fault trees, Reliability Engineering and System Safety, **58**:89-96.

Xu, K., Tang, L. C., Xie, M., Ho, S.L., Zhu, M. L., (2002): Fuzzy assessment of FMEA for engine systems, Reliability Engineering and System Safety,**75**:17-29.